《DSP 嵌入式系统开发典型案例》

作者：华清远见

第 4 章　常用自动控制系统设计

本章简介

随着自动化技术、计算机技术、集成芯片制造技术的飞速发展，自动控制系统的设计、实现也出现了飞跃式的发展，从单输入单输出系统发展到多输入多输出系统，从基本的 PID（Proportional Integral Differential）控制发展到目前种类繁多的最优控制、鲁棒控制、非线性控制、模糊控制、神经网络控制、滑模控制以及多种控制方法的结合控制技术，从自动控制系统的硬件实现来看，从 20 世纪 60 年代的分立元器件到 20 世纪 70 年代的中、小规模集成电路、再到 80 年代流行的以单片机为核心的数字化自动控制系统，然后到目前的以 DSP（数字信号处理器）为核心的高速、精密、智能的自动控制系统。

自动控制技术几乎应用于所有的工业部门，由于工业现场的工作环境、工作内容、控制对象、执行设备、动力设备、工作指标各不相同，因此工业控制的设计方法和实现手段也非常多，归纳起来，工业控制一般包括如下类别：过程控制、运动控制、速度伺服控制、位置伺服控制、点对点（I/O）控制等。过程控制和点对点（I/O）控制相对速度较慢，而运动控制和速度伺服控制以及位置伺服控制相对速度较快。

本章将基于 TI 公司 LF2407 和外接 DA 转换芯片，实现数字 PID 控制器，采用的 PID 控制算法是增量式 PID 控制算法，然后在此硬件平台上，以增量式 PID 控制算法为基础，介绍基于 LF2407 的模糊 PI 控制器的实现。

4.1 案例要求和应用对象

在过程控制、运动控制、速度伺服控制、位置伺服控制中，用得最多的控制方法是数字 PID 控制算法，目前虽然有很多优秀的控制方法都采用较新的控制算法，这些算法包括：积分分离 PID 控制算法、微分先行 PID 控制算法、带死区（dead-line）的 PID 算法、模糊 PID 算法等，但是这些算法都是建立在普通的数字 PID 控制算法的基础上的，普通的 PID 控制算法是其他一切控制算法的基础。虽然当前控制理论和控制技术在信息技术、集成电路制造技术的高速发展的推动下有了很大的发展，例如自适应控制、神经网络和模糊控制、鲁棒控制、滑模控制以及最优控制等很多现代控制方法都得到广泛的应用，但是数字 PID 控制仍然是一种稳定的、可靠的、实现简单的、使用广泛的控制方法。

PID 控制是技术最成熟的一种控制方法，特别是在控制对象的模型未知或难以建立时，常常采用 PID 控制方法。PID 控制原理简单、实现方便，并且适应性广、鲁棒性强，其控制品质对被控对象特性的变化不是很敏感。随着计算机技术的发展，在 PID 控制的基础上，出现了很多改进的数字 PID 控制方法，如微分先行 PID 控制、积分分离 PID 控制、带死区的 PID 控制（非线性 PID 控制）等。对于数字 PID 控制方法，又分为增量式 PID 控制算式和位置式 PID 控制算式。

本案例在 DSP 内部设置参考输入量，通过 LF2407 的片上 10bit AD 转换器采样，把被控对象的实际输出量采集到 DSP 中，经过 DSP 的数字运算处理后，通过外部的 DA 转换芯片（AD7237）进行数/模转换，得到实际的模拟控制量去控制被控对象，使之按照系统的设置运行工作。

另外本案例将在数字 PID 控制器的基础上，介绍基于 LF2407 的模糊 PI 控制器的实现。虽然数字 PID 控制器算法简单，不需要数学模型，且稳定性好、可靠性高。如果使用同样系数的 PID 控制器，一旦被控对象的工况发生变化，其控制品质将随之下降。也就是说如果在被控对象空载的情况下设定了 PID 控制器的各个参数，而当被控对象的温度、所加负载以及速度变化时，PID 控制器的控制品质将会下降。为此，控制理论提出了模糊 PI 控制器。模糊 PI 控制器是在 PI 控制器的基础上改进的，它的参数 K_p、K_i 将由模糊控制规则和专家经验决定。当被控对象的特性变化时，模糊 PI 控制器的参数 K_p、K_i 也随着发生变化，从而使得控制品质在各种不同的工况下均有良好的指标，满足工业控制的要求。

本案例介绍的数字 PID 控制器和模糊 PI 控制器的用途非常广泛，可以用于大多数工业控制现场，但不能应用于生物医疗设备、仪器的研制，因为采用本案例介绍的数字 PID 控制器和模糊 PI 控制器为了获得较快的动态过程，在状态变化的过渡过程中存在超调，而且对超调的抑制仅仅是采用微分环的作用，而模糊 PI 控制器中干脆就没有微分环节，这在工业生产部门是可以容忍的，而在生物医疗设备中却万万不能出现，生物医疗设备的自动控制设计的一个显著特点就是要抑制超调，而情愿牺牲其他的动态指标。

把本案例介绍的数字 PID 控制器和模糊 PI 控制器应用于工业控制现场，其动态指标的各方面比普通的控制器均有所提高，详细的指标评价在案例总结中有论述。图 4.1 是一个典型的采用数字 PID 控制器的，基于 LF2407 DSP 的速度控制系统的结构示意图。

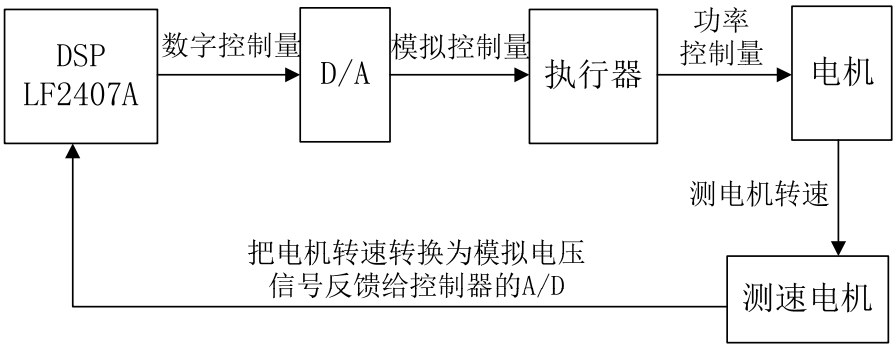


图 4.1 基于 LF2407 DSP 的速度控制系统的结构示意图

如图 4.1 所示，系统由 DSP LF2407A 作为控制器，由它产生一个数字控制量送入 DA 转换器，此处 DA 转换芯片采用 AD7237，由它进行数/模转换，得到模拟控制量，送入执行器，由执行器产生具有一定电流的功率控制量去直接控制电机，以上是一个开环的过程。自动控制系统中采用测速电机来测被控电机的转速，然后转换为模拟电压信号反馈给控制器 LF2407A 的片上外设——AD 转换器。另外需要注意的是，测速

电机种类繁多，根据不同的测速指标，测速电机的价格差别也很大，有的测速电机需要上万元，而普通的测速电机只有几百元甚至几十元，设计者一定要根据自身系统设计的需要购买，否则将大大增加系统的成本，难以推向市场。

4.2 硬件电路设计

基于 LF2407 的数字 PID 控制器和模糊 PI 控制器的 DSP 实现电路原理图是一样的，如图 4.2 所示。

从图 4.2 中可以看出，模拟量输入，也即被控电机的实际输出转速，通过一个限流电阻 R1 送到由 AD8041 构成的电压跟随器里，经过放大后，再经二极管 D1 和 D2 嵌位以及电阻 R2 限流，送到 LF2407 的片上 AD 转换器中，使得 AD 的输入限制在 0mV～3300mV。LF2407 一共有 16 路复用的片上 AD 转换器，本实例只采用第 0 通道的 AD 转换。

由 DSP 计算出控制量，通过外部数据总线送给 DA 转换器，采用 AD 公司的 AD7237 芯片，它是 8 位 DA 转换器，由 DSP 的外部地址总线 A2、A1、A0 构成 DA 转换器的译码电路，DA 转换器的输出采用 AD7237 的 A 相输出。

LF2407 由外部提供复位信号，由无源晶振提供 6MHz 时钟。片上 AD 转换器的参考电压高电平端 V_{refhi} 接 DSP 电源+3.3V，参考电压低电平端 V_{reflo} 接 DSP 的地。

TI 公司的 TMS320LF24xx 系列 DSP 是一种定点型 DSP，它的功能强大的 CPU 为 C2000 DSP 的应用提供了低成本、低功耗、高性能的处理能力，并且该芯片集成了多种集成外设，对工业应用中的电机数字化控制非常有用，是电机数字化控制的升级换代的产品。该系列 DSP 具有如下一些优点。

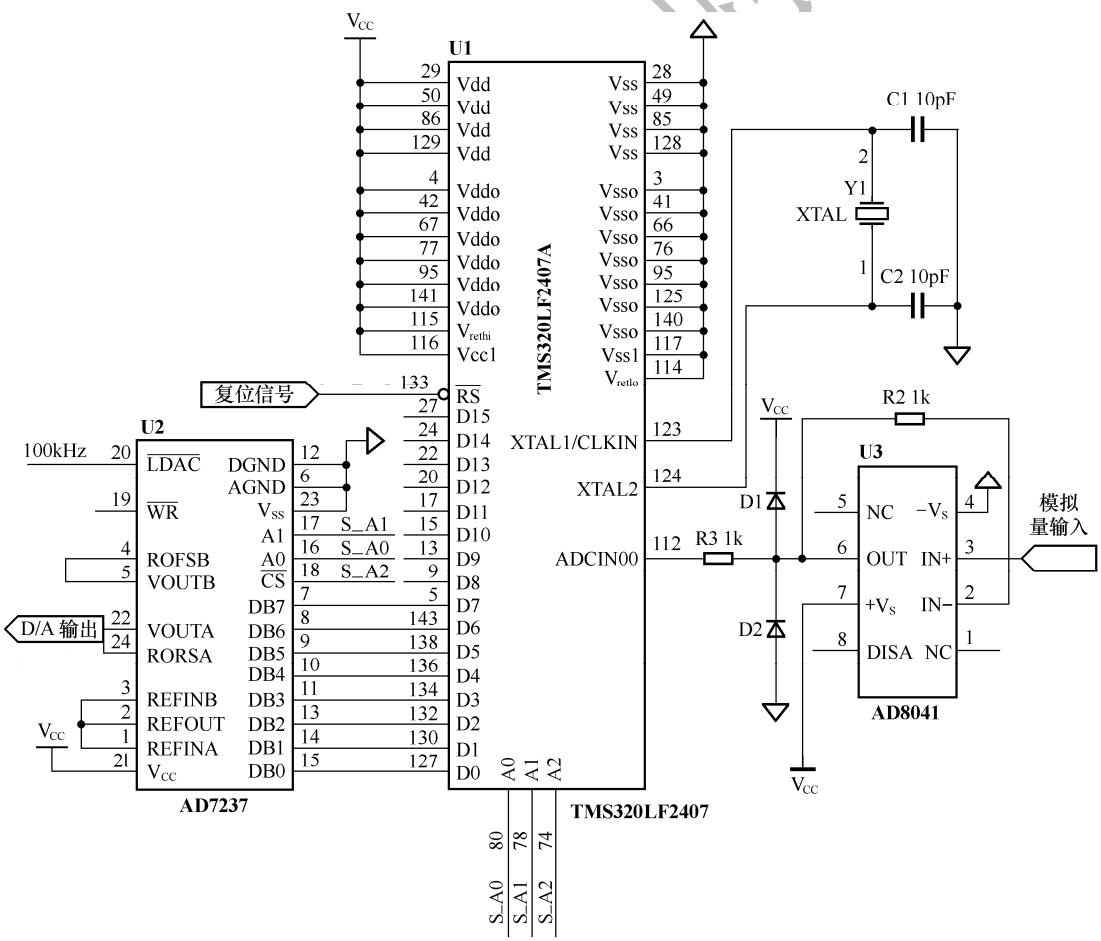


图 4.2 用 DSP 实现数字 PID 控制器和模糊 PI 控制器的电路原理图

➢ 采用高性能静态 CMOS 技术，使得供电电压为 3.3V，减小了控制器的功耗，另外 30MIPS 的执行速度使得指令周期缩短到 33ns（30MHz），从而提高了控制器的实时控制能力。

➢ 片内有高达 32KB 的 Flash 程序存储器，高达 1.5KB 的数据/程序 RAM，544Byte 的双口 RAM（DARAM）和 2KB 的单口 RAM（SARAM）。

➢ 两个事件管理器模块 EVA 和 EVB，每个都包括：2 个 16 位通用定时器；8 个 16 位的脉宽调制（PWM）通道。它们能够实现：三相反相器控制；PWM 的对称和非对称波形；当外部引脚/PDPINTx 出现低电平时快速关闭 PWM 通道；可编程的 PWM 死区控制以防止上下桥臂同时输出触发脉冲；3 个捕获单元；片内光电编码器接口电路；16 通道 AD 转换器。事件管理器模块适用于控制交流感应电机、无刷直流电机、开关磁阻电机、步进电机等。

➢ 可扩展的外部存储器最大共有 192KB 空间；64KB 程序存储器空间；64KB 数据存储器空间；64KB I/O 寻址空间。

➢ 看门狗定时器模块（WDT）。

➢ 10 位 AD 转换器的最小转换时间是 500ns，可选择由两个事件管理器来触发两个 8 通道输入的 AD 转换器或者一个 16 通道输入的 AD 转换器。

➢ 控制器局域网络（CAN）2.0 B 模块。

➢ 串行通信接口（SCI）模块。

➢ 16 位的串行外设（SPI）接口模块。

➢ 基于锁相环的时钟发生器。

➢ 高达 40 个可单独编程或复用的通用输入/输出引脚（GPIO）。

➢ 5 个外部中断，其中 2 个电机驱动保护、复位和 2 个可屏蔽中断。

➢ 电源管理包括 3 种低功耗模式，能独立地把外设器件转入低功耗工作状态。

本案例选用的 TI 公司的 TMS320LF24xx 系列的 DSP——TMS320LF2407A，它的 PGE 形式的封装图如图 4.3 所示。

另外一个涉及到系统性能指标的关键芯片是 DA 转换芯片，本案例中选用美国 AD 公司的 AD7237 来进行数/模转换，它的原理结构示意图如图 4.4 所示。AD7237 具有如下的一些关键特点。

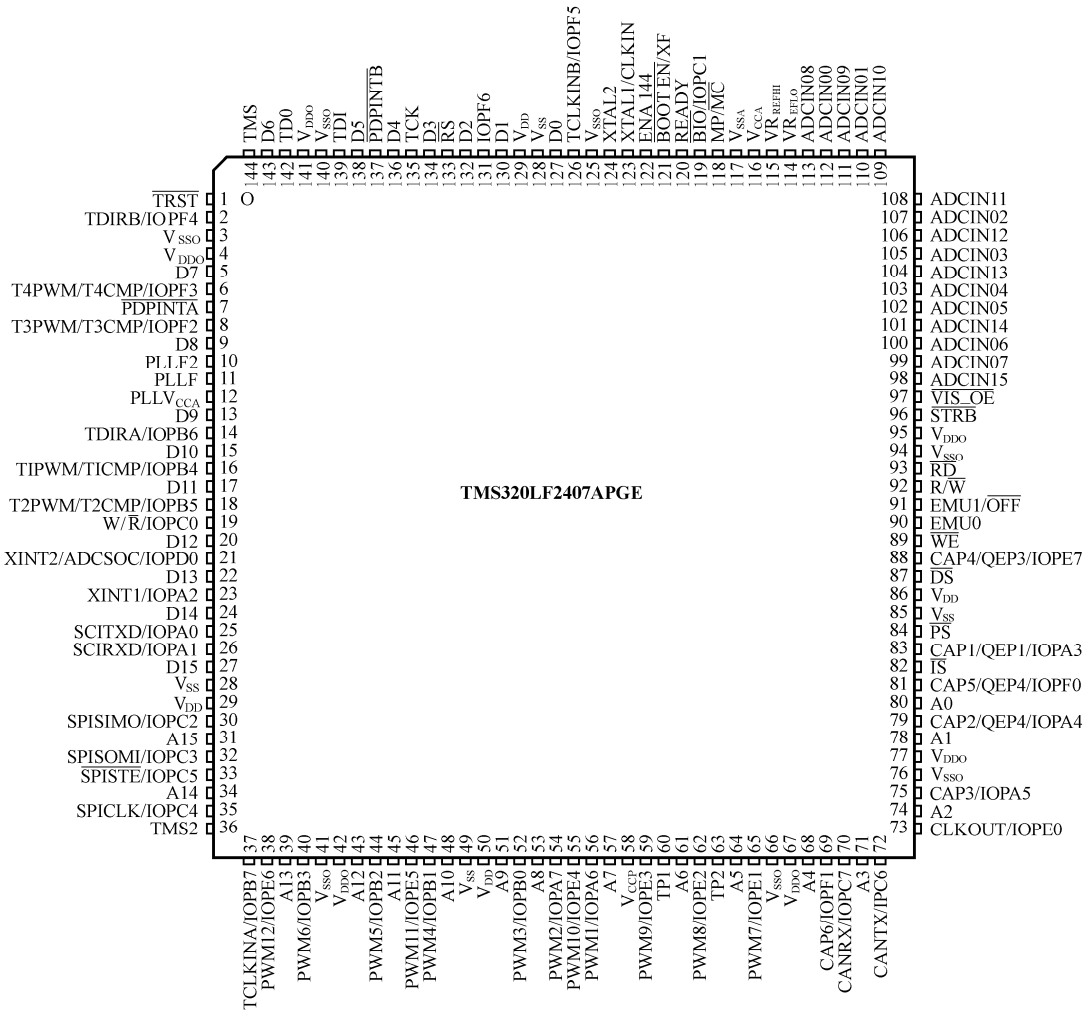


图 4.3　TMS320LF2407A DSP 的 PGE 形式的封装图

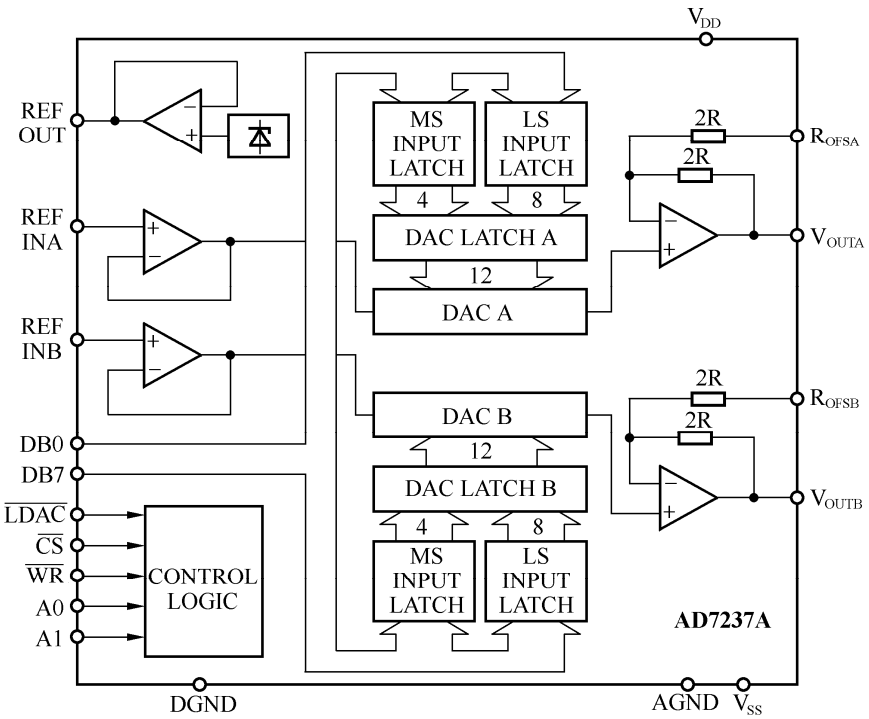


图 4.4　AD7237 的原理结构示意图

➢ 具有完全的双 12 位 DA 转换通道，可以提高整个系统的可靠性，改善系统的稳定性，特别适用于小结构的嵌入式系统。

➢ 片上具有电压参考输入。

➢ DA 转换的输出端具有放大电路，带载能力强。

➢ 芯片的操作电压是 12～15V，可以双电源供电，也可以是单电源供电。

➢ DA 转换输出的典型的数据建立时间是 30ns。

➢ 单电源供电时，典型的芯片功耗是 165mW。

如图 4.2 所示的硬件电路中，AD8041 构成了电压跟随器，对输入信号放大，再经二极管 D1 和 D2 嵌位以及电阻 R2 限流，送到 LF2407 的片上 AD 转换器中，本案例选用美国 AD 公司的 AD8041，它具有如下的一些关键特点。

➢ 多种电源形式供电，可以是+3V 或+5V 单电源供电，也可以是±5V 双电源供电，兼容性好。

➢ 片上提供轨到轨（Rail to Rail）的输出。

➢ 输入电压范围大，容故障能力强，在低于地电平 200mV 的输入电压下能正常工作。

➢ 片上具有/DISABLE 引脚，能够使芯片进入低功耗模式。

➢ 转换速度快，可以输入高频信号，其斜率为 160V/　s。

➢ 高频低失真，在 10MHz 频率下，最差谐波是－69 dBc。

➢ 输出端的带载能力强。

4.3　软件系统设计思路

在数字 PID 控制器和模糊 PI 控制器的程序设计中需要用到大量的变量，为了便于以后的讲解，首先定义一些程序中所要使用的变量名和公式中使用的变量名，如表 4.1 所示。

表 4.1　　公式中变量和程序中变量的对应关系及意义表

公式中的变量	意义描述	程序中的对应变量
$r(k)$	PID 控制器的参考输入量	PID_input
$c(k)$	PID 控制器的实际输入量	PID_reference
$u(k)$	PID 控制器的当前控制量	PID_output

$u(k-1)$	PID 控制器的上次控制量	PID_output1
$e(k)$	当前偏差量	PID_e0
$e(k-1)$	上次偏差量	PID_e1
$e(k-2)$	上上次偏差量	PID_e2
K_p	比例增益系数	Kp
K_i	积分增益系数高位字	Ki_high
K_i	积分增益系数低位字	Ki_low

续表

公式中的变量	意义描述	程序中的对应变量
K_d	微分增益系数	Kd
A	A 系数高位=$Kp+Ki+Kd$	A_coeff_high
A	A 系数低位=$Kp+Ki+Kd$	A_coeff_low
B	B 系数=$Kp+2\times Kd$	B_coeff
C	C 系数=Kd	用 Kd 代替

本案例首先设计一个数字 PID 控制器，现在假设它是一个对电动机速度进行 PID 控制的系统。图 4.5 是 PID 控制器的原理框图。

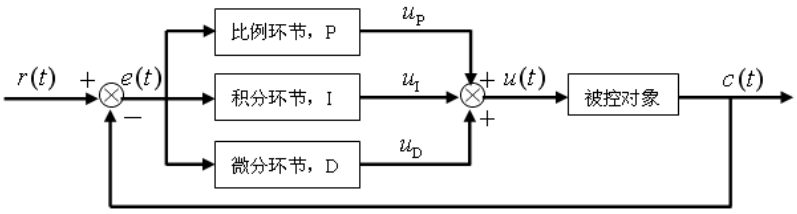


图 4.5　PID 控制器的原理框图

图 4.5 中，$r(t)$ 是电机速度设定值，$c(t)$ 是电机转速的实际测量值，$e(t)$ 是输入控制器的偏差信号，$u(t)$ 是控制器输出的控制量，则 PID 控制算式如式 4-1 所示。

$$u = K_p e + \frac{K_p}{T_i}\int_0^t e\mathrm{d}t + K_p T_d \frac{\mathrm{d}e}{\mathrm{d}t} \tag{4-1}$$

在式 4-1 中，K_p 是比例系数，起比例调整作用。T_i 是积分时间常数，它决定了积分作用的强弱。T_d 是微分时间常数，它决定了微分作用的强弱。在 PID 控制的 3 种作用中，比例作用可对系统的偏差做出及时响应；积分作用主要用来消除系统静差，改善系统的静态特性，体现了系统的静态性能指标；微分作用主要用来减少动态超调，克服系统振荡，加快系统的动态响应，改善系统的动态特性。PID 控制的 3 种作用（比例、积分、微分）是各自独立的，可以分别使用，也可以结合使用，但是积分控制和微分控制不能单独使用，必须和比例控制结合起来，形成 PI 控制器或者 PD 控制器。式 4-1 是模拟形式的 PID 控制算式，现在采用 LF2407 实现数字 PID 控制，则对式 4-1 离散化，得到 PID 控制的离散形式，如式 4-2 所示。

$$u(k) = K_p e(k) + \frac{K_p T_s}{T_i}\sum_{i=0}^{k} e(i) + K_p T_d \frac{e(k)-e(k-1)}{T_s} \tag{4-2}$$

其中 T_s 为采样周期。这是位置式 PID 控制算式，为了增加控制系统的可靠性，采用增量式 PID 控制算式，即让 LF2407 只输出控制量 $u(k)$ 的增量 $\Delta u(k)$。式 4-2 是第 k 次 PID 控制器的输出量，那么第 $k-1$ 次 PID 控制器的输出量如式 4-3 所示。

$$u(k-1)=K_{\mathrm{p}}e(k-1)+\frac{K_{\mathrm{p}}T_{\mathrm{s}}}{T_{\mathrm{i}}}\sum_{i=0}^{k-1}e(i)+K_{\mathrm{p}}T_{\mathrm{d}}\frac{e(k-1)-e(k-2)}{T_{\mathrm{s}}} \quad (4\text{-}3)$$

所以增量式 PID 控制算式如式 4-4 所示。

$$\begin{aligned}\Delta u(k)&=u(k)-u(k-1)\\&=K_{\mathrm{p}}[e(k)-e(k-1)]+\frac{K_{\mathrm{p}}T_{\mathrm{s}}}{T_{\mathrm{i}}}e(k)+K_{\mathrm{p}}T_{\mathrm{d}}\frac{e(k)-2e(k-1)+e(k-2)}{T_{\mathrm{s}}}\\&=(K_{\mathrm{p}}+K_{\mathrm{i}}+K_{\mathrm{d}})e(k)+(-K_{\mathrm{p}}-2K_{\mathrm{d}})e(k-1)+K_{\mathrm{d}}e(k-2)\end{aligned} \quad (4\text{-}4)$$

现令式 4-4 中的系数，

$A=(K_{\mathrm{p}}+K_{\mathrm{i}}+K_{\mathrm{d}})$，$-B=-(K_{\mathrm{p}}+2K_{\mathrm{d}})$，$C=K_d$。

则最后结果形式如方程式 4-5 所示。

$$\begin{cases}\Delta u(k)=A\cdot e(k)-B\cdot e(k-1)+C\cdot e(k-2)\\u(k)=u(k-1)+\Delta u(k)\end{cases} \quad (4\text{-}5)$$

方程式 4-5 就是本控制程序中用到的增量式 PID 控制算式。增量式 PID 控制与位置式 PID 控制相比仅是算法上有所改变，但是它只输出增量，减少了 DSP 误操作时对控制系统的影响，而且不会产生积分失控。本案例基于 LF2407 的 PID 控制器的实现框图如图 4.6 所示。

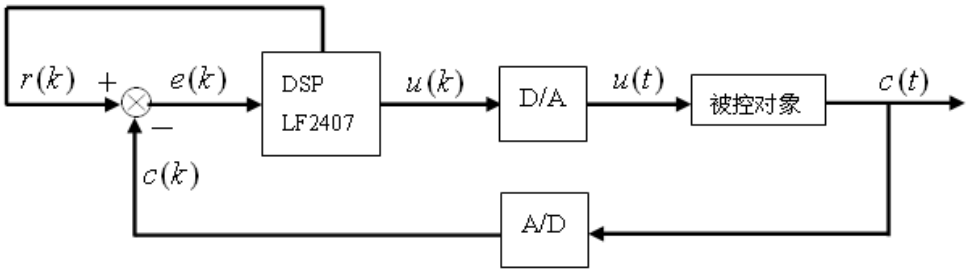


图 4.6　基于 LF2407 的 PID 控制器的实现框图

从图 4.6 可以看出被控电机的速度设定量由 DSP 给出，经过 DSP 计算出控制量 $u(k)$，对它进行 DA 转换，产生模拟控制量 $u(t)$，从而实现对被控电机速度的控制，而电机实际转速 $c(t)$ 通过 AD 转换器送入 DSP，使整个系统构成一个闭环系统。如图 4.7 所示是本案例设计的数字 PID 控制器在 DSP 上实现的控制程序流程图。

其次，本案例还要设计一个模糊 PI 控制器，它的硬件电路如图 4.2 所示，和数字 PID 控制器的硬件电路是一样的。模糊控制技术是建立在模糊数学的基础上的，它是针对被控对象的数学模型不明确，或非线性模型的一种工程实用、实现简单的控制方法。与传统的 PID 控制器相比，模糊控制器有更快的响应和更小的超调，对过程参数的变化不敏感，即具有很强的鲁棒性，能够克服非线性因素的影响。

数字 PID 控制器是一种工业控制中通用的控制器，但是工业生产现场环境复杂，往往出现在某种情况下设计好的控制参数，在另一种情况下又不满足工业生产的需求了，而常规的数字 PID 控制器不具有在线整定参数 K_p、K_i、K_d 的功能，使得其不能满足系统在不同偏差绝对值 $|e|$ 及偏差变化率绝对值 $|\triangle e|$ 下，对 PID 参数的不同要求，从而影响了控制器控制品质的进一步提高。在本案例中，在数字 PID 控制器的基础上，去掉数字 PID 控制器中的微分环节 D，只采用 PI 控制器，并且采用模糊推理思想，根据不同的 $|e|$ 和 $|\triangle e|$，对 PI 控制器的参数 K_p 和 K_i 进行在线自整定。其原理结构由两部分组成：常规 PI 控制器部分和模糊控制的参数校正部分，如图 4.8 所示。

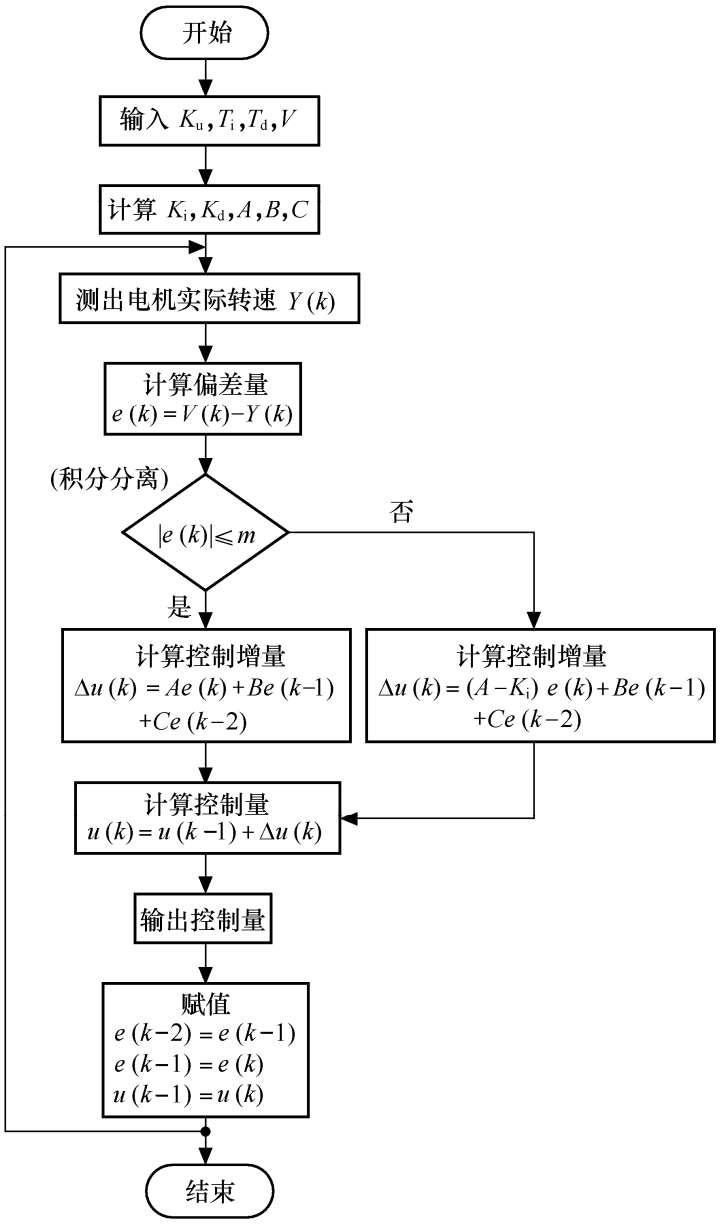


图 4.7　基于 LF2407 的数字 PID 控制器的程序流程图

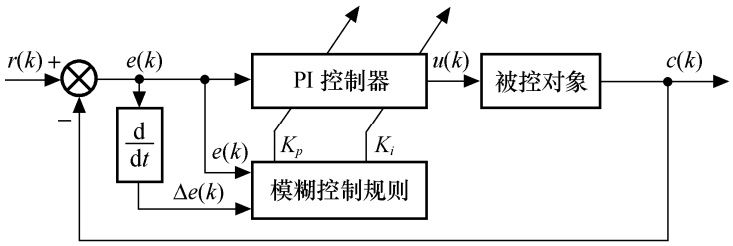


图 4.8　模糊 PI 控制器的原理框图

从图 4.8 中可以看出，$r(k)$ 是输入设定值，$c(k)$ 是实际测量值，$e(k)$ 是输入偏差信号，其 PI 控制器的离散表示形式如式 4-6 所示。

$$u(k)=K_{\mathrm{p}}e(k)+\frac{K_{\mathrm{p}}T_{\mathrm{s}}}{T_{\mathrm{i}}}\sum_{i=0}^{k}e(i) \tag{4-6}$$

在式 4-6 中，T_s 为采样周期。这是位置式 PI 控制算式，为了增加可靠性，采用增量式 PI 控制算式，如式 4-7 所示。增量式 PI 控制与位置式 PI 控制相比，仅仅是算法上有所改变，但是它只输出增量，减少了 DSP 误操作时对系统的影响。

$$\begin{aligned}\Delta u(k) &= u(k)-u(k-1)\\ &= K_{\mathrm{p}}[e(k)-e(k-1)]+\frac{K_{\mathrm{p}}T_{\mathrm{s}}}{T_{\mathrm{i}}}e(k)\\ &= (K_{\mathrm{p}}+K_{\mathrm{i}})e(k)-K_{\mathrm{p}}e(k-1)\end{aligned} \tag{4-7}$$

在式 4-7 中，令 A 系数=$K_{\mathrm{p}}+K_{\mathrm{i}}$，$B$ 系数=K_{p}，则最终增量式 PI 控制器的控制算式如方程式 4-8 所示。程序中只需计算出 A 系数和 B 系数就可以计算出当前的控制增量△$u(k)$。

$$\begin{cases}\Delta u(k) = A\cdot e(k)-B\cdot e(k-1)\\ u(k) = u(k-1)+\Delta u(k)\end{cases} \tag{4-8}$$

如图 4.8 所示，采用模糊控制规则，根据不同的$|e|$和$|\triangle e|$，对 P_{i}控制器的参数 K_{p}和 K_{i}进行在线自整定。本模糊控制器的输入语言变量是偏差绝对值$|E|$、偏差变化率绝对值$|\triangle E|$，输出语言变量是 P_{i}控制器的比例增益系数 K_{p}和积分增益系数 K_{i}。

在模糊控制中，语言变量是用语言值（模糊量）来表示一个物理量的，而不是用符号或确定的数字来表示的。当用语言值表示一个语言变量时，应注意用多少个语言值去描述语言变量，这是语言变量的分档问题。本实例中，各语言变量用语言值描述如下：

系统偏差的绝对值$|e(k)|$={零、小、中、大}

偏差变化率的绝对值$|\Delta e(k)|$={零、小、中、大}

P_{i}控制器的比例系数 K_{p} ={零、小、中、大}

P_{i}控制器的积分系数 K_{i} ={零、小、中、大}

各语言变量用相同的语言值来描述，简化了模糊控制规则的设计。在设计模糊控制规则时，采用不同的模糊推理，语言变量的分档是有区别的。本设计采用 CRI（Compositional Rule of Inference）推理法，为了在实时控制中避免进行关系矩阵的合成运算，先在脱机状态下把所有可能的输入和输出情况计算出来，形成一张控制表去执行控制。控制表是以整数形式表示的，为了能产生控制表，在 CRI 推理法中把语言变量的论域转换成有限整数的论域，本质上是把连续论域离散化后产生离散论域。采用式 4-9 把它离散化到整数论域 N。

$$\begin{cases}b = q\left(a-\dfrac{x_{\mathrm{L}}+x_{\mathrm{H}}}{2}\right)\\ q = \dfrac{2n}{x_{\mathrm{H}}-x_{\mathrm{L}}}\end{cases} \tag{4-9}$$

其中 a 为连续论域 $X=[x_{\mathrm{L}},x_{\mathrm{H}}]$中的某个数，$b$ 是与 a 对应的整数论域 N中的某个数，q 为模糊控制中对精确量进行模糊化时所用的量化因子。本设计中，各语言变量的档数均为 4 档（零、小、中、大），因此取整数论域 N为{0，1，2，3，4，5，6}。此时，如图 4.9 和图 4.10 所示，可以取语言变量值 4 档如下：

大（L）取在 5、6 附近；

中（M）取在 3、4 附近；

小（S）取在 1、2 附近；

零（Z）取在 0 附近。

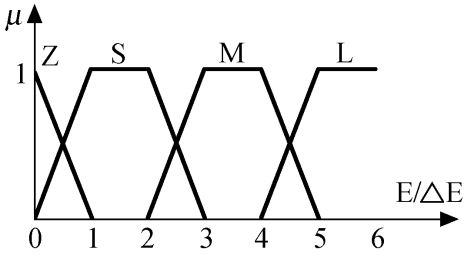


图 4.9 语言变量|E|和|△E|

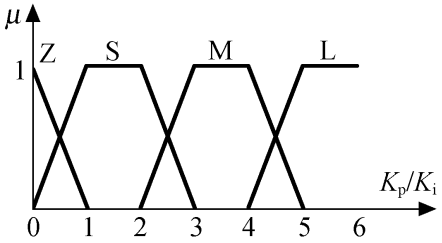


图 4.10 语言变量 K_p、K_i

由模糊推理得到的控制表中的控制量也是一个模糊量，反模糊化是模糊化的逆向运算，当整数论域为 $N=[-n,+n]$，连续论域 $X=[x_L,x_H]$，可采用式 4-10 进行反模糊化处理。

$$\begin{cases} a=k[b+\dfrac{n(x_L+x_H)}{x_H-x_L}] \\ k=\dfrac{x_H-x_L}{2n} \end{cases} \tag{4-10}$$

其中 b 为整数论域为 $N=[-n,+n]$ 中的某个数，a 是与 b 对应的连续论域 $X=[x_L,x_H]$ 中的某个数，k 为模糊控制中对模糊量进行反模糊化时所用的比例因子。

当 $|e(k)|$ 较大时，为了加快系统的响应速度，并为避免因开始时偏差的瞬间变大可能引起微分过饱和而使控制作用超出许可范围，应取较大的 K_p，同时为了防止积分饱和，避免系统响应出现较大的超调，此时应该去掉积分作用，取 K_i =0。

当 $|e(k)|$ 和 $|\triangle e(k)|$ 为中等大小时，为使系统响应的超调减小又不影响系统的响应速度，K_p 和 K_i 都不能取大，而应取较小的 K_i，K_p 的取值要大小适中。

当 $|e(k)|$ 较小时，为使系统具有良好的稳态性能，应该增大 K_p 和 K_i 的值，同时为了避免系统在设定值附近出现振荡，并考虑到系统的抗干扰性能，应适当地选取 K_p。

根据以上这些原则，并参考工作中总结的实际经验，得到了如表 4.2 所示的模糊 PI 控制器参数 K_p 和 K_i 的调节规则，但是这些规则都是用模糊量来表示的。

表 4.2 参数自整定模糊 PI 控制规则表

K_p，K_i		$\|\triangle E(k)\|$			
		Z	S	M	L
$\|E(k)\|$	Z	Z, L	L, L	L, L	M, L
	S	L, L	L, L	L, L	M, M
	M	M, Z	M, Z	M, S	S, S
	L	L, Z	L, Z	L, Z	M, Z

在本设计中，利用 CRI 法推理时控制过程是用查控制表来产生控制量的。在控制表中，模糊偏差量|E|、模糊偏差变化率|△E|、P_i 控制器的模糊比例系数 K_p、模糊积分系数 K_i 都是用其对应整数论域的元素来表示的。对于单个实时精确量，把它和量化因子相乘，得到的结果再四舍五入，就求出了对应整数论域的相应元素，即可用于查询控制表，从而实现了输入量的模糊化。根据以上分析，如图 4.11 所示是本案例设计的模糊 PI 控制器在 DSP 上实现的控制程序流程图。

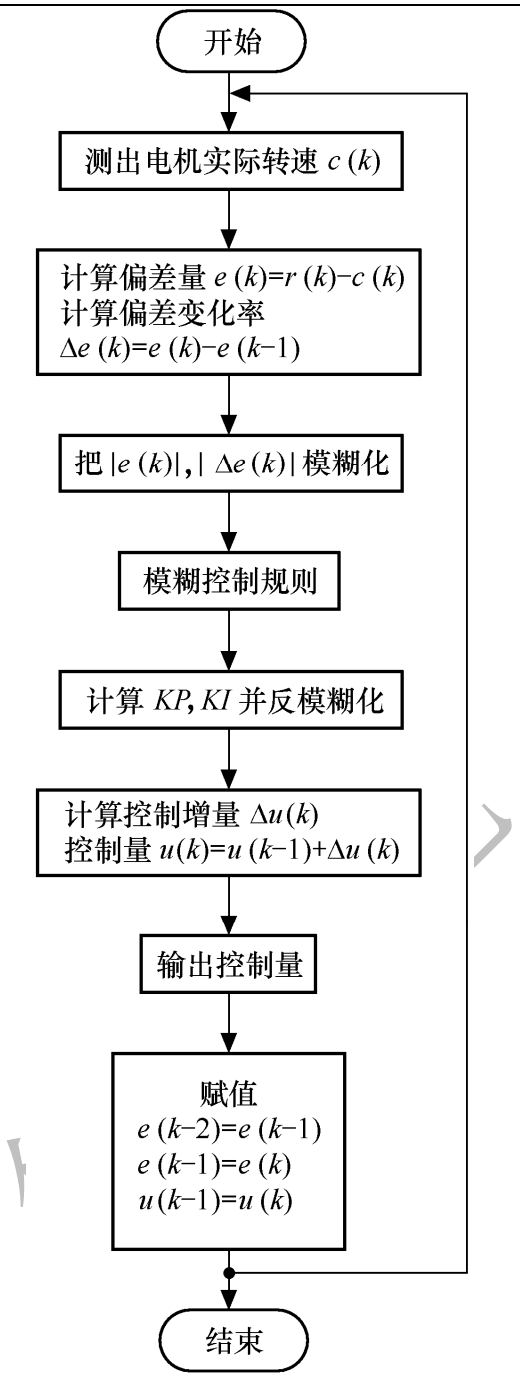


图 4.11　基于 LF2407 的模糊 PI 控制器的程序流程图

4.4　主要程序代码说明

实用的 DSP 程序包括主程序和系统初始化程序以及存储器配置文件、中断向量表程序，如果系统没有正确初始化或者存储器配置不正确，或者没有中断向量表，则系统将不能正确运行，得到的结果和预想的将大相径庭。本节将先介绍 DSP 的系统初始化、存储器配置和中断向量表，然后再介绍包含实际算法的 DSP 应用程序。

4.4.1　DSP 配置头文件

一般来说，不管是哪个公司生产的 DSP 芯片，它们都包括很多存储器映射的 CPU 寄存器和外设电路寄存器，它们在 DSP 的数据存储空间中的固定位置，在编程中需要经常对它们进行操作，因此需要编写一个

头文件，预先把这些寄存器的名称和它们在数据存储空间的地址一一对应起来。针对某款 DSP 芯片编写的头文件对所有的基于该款 DSP 的程序都是通用的，本案例是基于 TI 公司的 C2000 DSP 系列的 LF2407 开发的，所以头文件 lf2407_regs.h 就是 LF2407A 的所有寄存器的配置文件，具体程序如下所示，由于 DSP 片上外设丰富，涉及到的寄存器也非常多，因此在头文件中对所有片上的寄存器都进行定义是非常必要的。本程序对每个寄存器都做了详细介绍，在今后对 LF2407 进行开发时，只要把该文件用汇编伪指令.include 包括到主程序中即可使用。

```
;*****************************************************************
;;  文件名：lf2407_regs.h
;;  功能描述：lf240x/240xA 的 CPU 和外设寄存器以及其他常用定义
;*****************************************************************
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
; 全局变量寄存器和 CPU 中断寄存器
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
IMR           .set 0004h          ; 中断屏蔽寄存器
GREG      .set 0005h          ; 全局变量寄存器
IFR           .set 0006h          ; 中断标志寄存器
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ ~~~~~~~~~~~~~~~~~~~~
; 系统配置和控制寄存器
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
PIRQR0        .set 07010h         ; 外设中断请求寄存器 0
PIRQR1        .set 07011h         ; 外设中断请求寄存器 1
PIRQR2        .set 07012h         ; 外设中断请求寄存器 2
PIACKR0       .set 07014h         ; 外设中断应答寄存器 0
PIACKR1       .set 07015h         ; 外设中断应答寄存器 1
PIACKR2       .set 07016h         ; 外设中断应答寄存器 2
SCSR1         .set 07018h         ; 系统控制和状态寄存器 1
SCSR2         .set 07019h         ; 系统控制和状态寄存器 2
DINR      .set 0701Ch         ; 系统模块状态寄存器
PIVR          .set 0701Eh         ; 外设中断矢量寄存器
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
; 程序监视控制寄存器(看门狗)
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
WDCNTR    .set 07023h         ; 看门狗计数器寄存器
WDKEY         .set 07025h         ; 看门狗复位密钥寄存器
WDCR      .set 07029h         ; 看门狗控制寄存器
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
; 串行外设接口 (SPI) 寄存器
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
SPICCR        .set 07040h         ;  SPI 配置和控制寄存器
SPICTL        .set 07041h         ;  SPI 操作控制寄存器
SPISTS        .set 07042h         ;  SPI 状态寄存器
SPIBRR        .set 07044h         ;  SPI 波特率寄存器
SPIEMU        .set 07046h         ;  SPI 仿真缓冲寄存器
SPIRXBUF  .set 07047h         ;  SPI 串行接收缓冲寄存器
SPITXBUF  .set 07048h         ;  SPI 串行发送缓冲寄存器
SPIDAT        .set 07049h         ;  SPI 串行数据寄存器
SPIPRI        .set 0704Fh         ;  SPI 中断优先级控制寄存器
```

```
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
; 串行通信接口 (SCI) 寄存器
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
SCICCR        .set 07050h        ;   SCI 通信控制寄存器
SCICTL1       .set 07051h        ;   SCI 控制寄存器 1
SCIHBAUD .set 07052h        ;   SCI 波特率寄存器高位
SCILBAUD .set 07053h        ;   SCI 波特率寄存器低位
SCICTL2       .set 07054h        ;   SCI 控制寄存器 2
SCIRXST       .set 07055h        ;   SCI 接收状态寄存器
SCIRXEMU .set 07056h        ;   SCI 仿真数据缓冲寄存器
SCIRXBUF .set 07057h        ;   SCI 接收数据缓冲寄存器
SCITXBUF .set 07059h        ;   SCI 发送数据缓冲寄存器
SCIPRI        .set 0705Fh        ;   SCI 中断优先级控制寄存器
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
; 外部中断寄存器
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
XINT1CR       .set 07070h        ; 中断 1 控制寄存器
XINT2CR   .set 07071h        ; 中断 2 控制寄存器
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
; 数据 I/O 控制寄存器
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
MCRA      .set 07090h        ;   I/O 口复用控制寄存器 A
MCRB      .set 07092h        ;   I/O 口复用控制寄存器 B
MCRC      .set 07094h        ;   I/O 口复用控制寄存器 C
PEDATDIR .set 07095h        ; 端口 E 的数据和方向控制寄存器
PFDATDIR .set 07096h        ; 端口 F 的数据和方向控制寄存器
PADATDIR .set 07098h        ; 端口 A 的数据和方向控制寄存器
PBDATDIR .set 0709Ah        ; 端口 B 的数据和方向控制寄存器
PCDATDIR .set 0709Ch        ; 端口 C 的数据和方向控制寄存器
PDDATDIR .set 0709Eh        ; 端口 D 的数据和方向控制寄存器
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
; 模数转换(ADC)寄存器
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
ADCCTRL1 .set 070A0h        ;   ADC 控制寄存器 1
ADCCTRL2 .set 070A1h        ;   ADC 控制寄存器 2
MAXCONV   .set 070A2h        ; 最大转换通道数寄存器
CHSELSEQ1.set 070A3h        ; 通道选择排序控制寄存器 1
CHSELSEQ2.set 070A4h        ; 通道选择排序控制寄存器 2
CHSELSEQ3.set 070A5h        ; 通道选择排序控制寄存器 3
CHSELSEQ4.set 070A6h        ; 通道选择排序控制寄存器 4
AUTO_SEQ_SR  .set  070A7h       ; 自动排序状态寄存器
RESULT0       .set 070A8h        ;   AD 转换结果寄存器 0
RESULT1       .set 070A9h        ;   AD 转换结果寄存器 1
RESULT2       .set 070AAh        ;   AD 转换结果寄存器 2
RESULT3       .set 070ABh        ;   AD 转换结果寄存器 3
RESULT4       .set 070ACh        ;   AD 转换结果寄存器 4
RESULT5       .set 070ADh        ;   AD 转换结果寄存器 5
```

```
RESULT6       .set 070AEh        ;   AD 转换结果寄存器 6
RESULT7       .set 070AFh        ;   AD 转换结果寄存器 7
RESULT8       .set 070B0h        ;   AD 转换结果寄存器 8
RESULT9       .set 070B1h        ;   AD 转换结果寄存器 9
RESULT10  .set 070B2h        ;   AD 转换结果寄存器 10
RESULT11  .set 070B3h        ;   AD 转换结果寄存器 11
RESULT12  .set 070B4h        ;   AD 转换结果寄存器 12
RESULT13  .set 070B5h        ;   AD 转换结果寄存器 13
RESULT14  .set 070B6h        ;   AD 转换结果寄存器 14
RESULT15  .set 070B7h        ;   AD 转换结果寄存器 15
CALIBRATION    .set     070B8h        ; 校准结果寄存器
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
; CAN 配置控制寄存器
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
MDER      .set 07100h        ; 邮箱方向/使能控制寄存器
TCR           .set 07101h        ; 发送控制寄存器
RCR           .set 07102h        ; 接收控制寄存器
MCR           .set 07103h        ; 主控制寄存器
BCR2      .set 07104h        ; 位定时器配置寄存器 2
BCR1      .set 07105h        ; 位定时器配置寄存器 1
ESR           .set 07106h        ; 错误状态寄存器
GSR           .set 07107h        ; 全局状态寄存器
CEC           .set 07108h        ;   CAN 错误计数寄存器
CAN_IFR       .set   07109h      ;   CAN 中断标志寄存器
CAN_IMR       .set   0710Ah      ;   CAN 中断屏蔽寄存器
LAM0_H        .set 0701Bh        ; 对于 MBOX0 和 1 的局部接收屏蔽高位寄存器
LAM0_L        .set 0701Ch        ; 对于 MBOX0 和 1 的局部接收屏蔽低位寄存器
LAM1_H        .set 0701Dh        ; 对于 MBOX2 和 3 的局部接收屏蔽高位寄存器
LAM1_L        .set 0701Eh        ; 对于 MBOX2 和 3 的局部接收屏蔽低位寄存器
;;;; 邮包#0
MSGID0L       .set 07200h        ; 邮箱标识符低位寄存器 0
MSGID0H       .set 07201h        ; 邮箱标识符高位寄存器 0
MSGCTRL0  .set 07202h        ; 邮箱控制寄存器 0
MBX0A         .set 07204h        ; 邮箱 0 寄存器 A
MBX0B         .set 07205h        ; 邮箱 0 寄存器 B
MBX0C         .set 07206h        ; 邮箱 0 寄存器 C
MBX0D         .set 07207h        ; 邮箱 0 寄存器 D
;;;;邮包#1
MSGID1L       .set 07208h        ; 邮箱标识符低位寄存器 1
MSGID1H       .set 07209h        ; 邮箱标识符高位寄存器 1
MSGCTRL1  .set 0720Ah        ; 邮箱控制寄存器 1
MBX1A         .set 0720Ch        ; 邮箱 1 寄存器 A
MBX1B         .set 0720Dh        ; 邮箱 1 寄存器 B
MBX1C         .set 0720Eh        ; 邮箱 1 寄存器 C
MBX1D         .set 0720Fh        ; 邮箱 1 寄存器 D
;;;; 邮包#2
MSGID2L       .set 07210h        ; 邮箱标识符低位寄存器 2
```

```
MSGID2H       .set 07211h        ; 邮箱标识符高位寄存器 2
MSGCTRL2  .set 07212h        ; 邮箱控制寄存器 2
MBX2A         .set 07214h        ; 邮箱 2 寄存器 A
MBX2B         .set 07215h        ; 邮箱 2 寄存器 B
MBX2C         .set 07216h        ; 邮箱 2 寄存器 C
MBX2D         .set 07217h        ; 邮箱 2 寄存器 D
;;;; 邮包#3
MSGID3L       .set 07218h        ; 邮箱标识符低位寄存器 3
MSGID3H       .set 07219h        ; 邮箱标识符高位寄存器 3
MSGCTRL3  .set 0721Ah        ; 邮箱控制寄存器 3
MBX3A         .set 0721Ch        ; 邮箱 3 寄存器 A
MBX3B         .set 0721Dh        ; 邮箱 3 寄存器 B
MBX3C         .set 0721Eh        ; 邮箱 3 寄存器 C
MBX3D         .set 0721Fh        ; 邮箱 3 寄存器 D
;;;; 邮包#4
MSGID4L       .set 07220h        ; 邮箱标识符低位寄存器 4
MSGID4H       .set 07221h        ; 邮箱标识符高位寄存器 4
MSGCTRL4  .set 07222h        ; 邮箱控制寄存器 4
MBX4A         .set 07224h        ; 邮箱 4 寄存器 A
MBX4B         .set 07225h        ; 邮箱 4 寄存器 B
MBX4C         .set 07226h        ; 邮箱 4 寄存器 C
MBX4D         .set 07227h        ; 邮箱 4 寄存器 D
;;;; 邮包#5
MSGID5L       .set 07228h        ; 邮箱标识符低位寄存器 5
MSGID5H       .set 07229h        ; 邮箱标识符高位寄存器 5
MSGCTRL5  .set 0722Ah        ; 邮箱控制寄存器 5
MBX5A         .set 0722Ch        ; 邮箱 5 寄存器 A
MBX5B         .set 0722Dh        ; 邮箱 5 寄存器 B
MBX5C         .set 0722Eh        ; 邮箱 5 寄存器 C
MBX5D         .set 0722Fh        ; 邮箱 5 寄存器 D
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
; 通用定时器 ——> 事件管理器 A (EVA)
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
GPTCONA   .set 7400h         ; 通用定时控制寄存器
T1CNT         .set 7401h         ; 通用定时器 1 计数寄存器
T1CMPR        .set 7402h         ; 通用定时器 1 比较寄存器
T1PR      .set 7403h         ; 通用定时器 1 周期寄存器
T1CON         .set 7404h         ; 通用定时器 1 控制寄存器
T2CNT         .set 7405h         ; 通用定时器 2 计数寄存器
T2CMPR        .set 7406h         ; 通用定时器 2 比较寄存器
T2PR      .set 7407h         ; 通用定时器 2 周期寄存器
T2CON         .set 7408h         ; 通用定时器 2 控制寄存器
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
; 比较单元的寄存器 ——> 事件管理器 A (EVA)
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
COMCONA   .set 7411h         ; 比较控制寄存器 A
ACTRA         .set 7413h         ; 比较方式控制寄存器 A
```

```
DBTCONA   .set 7415h          ; 死区控制寄存器 A
CMPR1          .set 7417h          ; 全比较单元 1 比较寄存器
CMPR2          .set 7418h          ; 全比较单元 2 比较寄存器
CMPR3          .set 7419h          ; 全比较单元 3 比较寄存器
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
; 捕捉和正交编码电路的寄存器 ——> 事件管理器(EVA)
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
CAPCONA   .set 7420h          ; 捕捉控制寄存器 A
CAPFIFOA  .set 7422h          ; 捕捉 FIFO 状态寄存器 A
CAP1FIFO       .set 7423h          ; 捕捉 1 二级 FIFO 寄存器
CAP2FIFO       .set 7424h          ; 捕捉 2 二级 FIFO 寄存器
CAP3FIFO       .set 7425h          ; 捕捉 3 二级 FIFO 寄存器
CAP1FBOT  .set 7427h          ; 捕捉 1 的 FIFO 栈的栈底寄存器
CAP2FBOT  .set 7428h          ; 捕捉 2 的 FIFO 栈的栈底寄存器
CAP3FBOT  .set 7429h          ; 捕捉 3 的 FIFO 栈的栈底寄存器
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
; 事件管理器 (EVA) 中断控制寄存器
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
EVAIMRA        .set 742Ch          ; 事件管理器中断屏蔽寄存器 A
EVAIMRB        .set 742Dh          ; 事件管理器中断屏蔽寄存器 B
EVAIMRC        .set 742Eh          ; 事件管理器中断屏蔽寄存器 C
EVAIFRA        .set 742Fh          ; 事件管理器中断标志寄存器 A
EVAIFRB        .set 7430h          ; 事件管理器中断标志寄存器 B
EVAIFRC        .set 7431h          ; 事件管理器中断标志寄存器 C
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
; 通用定时器 ——> 事件管理器 B (EVB)
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
GPTCONB   .set 7500h               ; 通用定时控制寄存器
T3CNT          .set 7501h               ; 通用定时器 3 计数寄存器
T3CMPR         .set 7502h               ; 通用定时器 3 比较寄存器
T3PR      .set 7503H               ; 通用定时器 3 周期寄存器
T3CON          .set 7504h               ; 通用定时器 3 控制寄存器
T4CNT          .set 7505h               ; 通用定时器 4 计数寄存器
T4CMPR         .set 7506h               ; 通用定时器 4 比较寄存器
T4PR      .set 7507H               ; 通用定时器 4 周期寄存器
T4CON          .set 7508h               ; 通用定时器 4 控制寄存器
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
; 比较单元寄存器 ——> 事件管理器 B (EVB)
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
COMCONB   .set 07511h              ; 比较控制寄存器 B
ACTRB          .set 07513h              ; 比较方式控制寄存器 B
DBTCONB   .set 07515h              ; 死区控制寄存器 B
CMPR4          .set 07517h              ; 全比较单元 4 比较寄存器
CMPR5          .set 07518h              ; 全比较单元 5 比较寄存器
CMPR6          .set 07519h              ; 全比较单元 6 比较寄存器
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
; 捕捉单元寄存器 ——> 事件管理器 B (EVB)
```

```
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
CAPCONB   .set 7520h              ; 捕捉控制寄存器 B
CAPFIFOB  .set 7522h              ; 捕捉 FIFO 状态寄存器 B
CAP4FIFO       .set 7523h              ; 捕捉 4 二级 FIFO 寄存器
CAP5FIFO       .set 7524h              ; 捕捉 5 二级 FIFO 寄存器
CAP6FIFO       .set 7525h              ; 捕捉 6 二级 FIFO 寄存器
CAP4FBOT  .set 7527h              ; 捕捉 4 的 FIFO 栈的栈底寄存器
CAP5FBOT  .set 7528h              ; 捕捉 5 的 FIFO 栈的栈底寄存器
CAP6FBOT  .set 7529h              ; 捕捉 6 的 FIFO 栈的栈底寄存器
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
; 事件管理器 (EVB) 中断控制寄存器
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
EVBIMRA        .set 742Ch         ; 事件管理器中断屏蔽寄存器 A
EVBIMRB        .set 742Dh         ; 事件管理器中断屏蔽寄存器 B
EVBIMRC        .set 742Eh         ; 事件管理器中断屏蔽寄存器 C
EVBIFRA        .set 742Fh         ; 事件管理器中断标志寄存器 A
EVBIFRB        .set 7430h         ; 事件管理器中断标志寄存器 B
EVBIFRC        .set 7431h         ; 事件管理器中断标志寄存器 C
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
; 程序存储器空间——FLASH 寄存器
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
PMPC      .set 0h         ; FLASH 段控制寄存器
CTRL      .set 1h
WADDR          .set 2h         ; FLASH 写地址寄存器
WDATA          .set 3h         ; FLASH 写数据寄存器
TCRB      .set 4h
ENAB      .set 5h
SETC      .set 6h
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
; I/O 存储空间
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
FCMR      .set 0FF0Fh     ; FLASH 控制模式寄存器
WSGR      .set 0FFFFh     ; 等待状态产生器控制寄存器
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
; 数据存储器块地址映射
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
B0_SADDR  .set 00200h          ; 块 B0 开始地址
B0_EADDR  .set 002FFh          ; 块 B0 结束地址
B1_SADDR  .set 00300h          ; 块 B1 开始地址
B1_EADDR  .set 003FFh          ; 块 B1 结束地址
B2_SADDR  .set 00060h          ; 块 B2 开始地址
B2_EADDR  .set 0007Fh          ; 块 B2 结束地址
XDATA_SADDR    .set 08000h     ; 外部数据空间开始地址
XDATA_EADDR    .set  0FFFFh    ; 外部数据空间结束地址
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
; 经常使用的数据页
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
DP_B2          .set      0               ; 页 0 数据空间
DP_B01         .set 4                ; 页 4  B0(200H/80H)
DP_B02         .set 5                ; 页 5  B0(280H/80H)
DP_B11         .set 6                ; 页 6  B1(300H/80H)
DP_B12         .set 7                ; 页 7  B1(380H/80H)

DP_PF1         .set      224           ; 页 1  外设帧文件  (7000h/80h)(0XE0)
DP_PF2         .set      225           ; 页 2  外设帧文件  (7080h/80h)(0XE1)
DP_PF3         .set      226           ; 页 3  外设帧文件  (7100h/80h)(0XE2)
DP_PF4         .set      227           ; 页 4  外设帧文件  (7180h/80h)(0XE3)
DP_PF5         .set      228           ; 页 5  外设帧文件  (7200h/80h)(0XE4)
DP_EVA         .set 232           ; 页 0 事件管理器-EVA 文件 (7400h/80h)(0xE8)
DP_EVB         .set 234           ; 页 0 事件管理器-EVB 文件 (7500h/80h)(0xEA)
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
; 位测试指令的位代码 (BIT)
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
BIT15          .set 0000h          ; 位代码 15
BIT14          .set 0001h          ; 位代码 14
BIT13          .set 0002h          ; 位代码 13
BIT12          .set 0003h          ; 位代码 12
BIT11          .set 0004h          ; 位代码 11
BIT10          .set 0005h          ; 位代码 10
BIT9           .set 0006h          ; 位代码 9
BIT8           .set 0007h          ; 位代码 8
BIT7           .set 0008h          ; 位代码 7
BIT6           .set 0009h          ; 位代码 6
BIT5           .set 000Ah          ; 位代码 5
BIT4           .set 000Bh          ; 位代码 4
BIT3           .set 000Ch          ; 位代码 3
BIT2           .set 000Dh          ; 位代码 2
BIT1           .set 000Eh          ; 位代码 1
BIT0           .set 000Fh          ; 位代码 0
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
; 常用屏蔽位
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
B15_MSK   .set 8000h             ; 位屏蔽 15
B14_MSK   .set 4000h             ; 位屏蔽 14
B13_MSK   .set 2000h             ; 位屏蔽 13
B12_MSK   .set 1000h             ; 位屏蔽 12
B11_MSK   .set 0800h             ; 位屏蔽 11
B10_MSK   .set 0400h             ; 位屏蔽 10
B9_MSK         .set 0200h             ; 位屏蔽 9
B8_MSK         .set 0100h             ; 位屏蔽 8
B7_MSK         .set 0080h             ; 位屏蔽 7
B6_MSK         .set 0040h             ; 位屏蔽 6
B5_MSK         .set 0020h             ; 位屏蔽 5
B4_MSK         .set 0010h             ; 位屏蔽 4
```

```
B3_MSK        .set 0008h            ; 位屏蔽 3
B2_MSK        .set 0004h            ; 位屏蔽 2
B1_MSK        .set 0002h            ; 位屏蔽 1
B0_MSK        .set 0001h            ; 位屏蔽 0
```

4.4.2 DSP 的系统配置命令文件

TI 公司开发的 DSP 汇编器和链接器所创建的目标文件采用公共目标文件格式（Common Object File Format，简称 COFF 文件），采用这种目标文件格式更利于模块化编程，并且为管理代码段和目标系统的存储器提供了更强有力和更加灵活的方法。基于 COFF 文件格式编写汇编程序或高级语言程序时，不必为程序代码或变量指定目标地址，这为程序编写和移植提供了极大的方便。COFF 文件格式鼓励程序员在用汇编语言或高级语言编程时基于代码块和数据块的概念，而不是一条条命令和一个个数据，这使得程序的可读性和可移植性大大增强。在 COFF 文件格式中，汇编器和链接器都提供了有关命令来创建块和对块进行处理。

链接器对块进行处理具有两个功能，首先它把 COFF 目标文件中的块用来建立程序块或数据块，它把输入块组合起来，以建立可执行的 COFF 输出模块。其次，链接器为输出块选择存储器地址，链接器提供了两个命令来完成上述功能：MEMORY 命令和 SECTIONS 命令。MEMORY 命令定义目标系统的存储器，程序员可以定义每一块存储器的起始地址和长度。SECTIONS 命令告诉链接器如何组合输入块以及在存储器的何处存放该输出块。因此，一个 DSP 程序正确运行离不开系统配置命令文件（*.cmd），该文件实现对程序存储空间和数据存储空间的分配。由于 DSP 的程序空间和数据空间是分开的，因此在 PAGE 0（程序空间）和 PAGE 1（数据空间）中的地址是可以重叠的，但必须保证在物理存储器上，它们是分离的。而在同一页存储空间上地址是不能重叠的。PAGE 2 表示系统的 I/O 存储空间，一般不使用，可以不在命令文件中列出。

下面的程序代码是本案例程序的系统配置命令文件，该文件实现对程序存储空间和数据存储空间的分配，从该配置文件中可以看出本案例的存储器资源和配置方法。另外需要注意的是，其他 C2000 的 DSP 的系统配置命令文件都可以参考本程序，不同的是程序空间和数据空间的分块和容量大小得根据具体的应用系统的要求来划分。

```
-stack 40       /* 设置堆栈长度 */
MEMORY     /* 声明在目标系统中实际存在的且可以被使用的存储器范围 */
{
     PAGE 0:                                  /* 程序空间 */
          VECS:     origin=0000h,length=0040h   /* 中断向量存储空间 */
          PVECS:    origin=0044h,length=0100h   /* 外围中断向量 */
             PM:    origin=0150h,length=7EB0h        /* 片内 Flash 存储空间 */

     PAGE 1:                                  /* 数据空间 */
          BLOCK_B2: origin=0060h,length=0020h   /* 块 B2 */
          BLOCK_B0: origin=0200h,length=0100h   /* 块 B0，如果 CNF=0，则分配为片内 DARAM */
          BLOCK_B1: origin=0300h,length=0100h   /* 块 B1 */
/* 如果正确配置 SCSR2 寄存器，则此为数据空间中 2K 的 SARAM */
SARAM:    origin=0800h, length=0800h
          EX_DM:    origin=8000h, length=8000h       /* 片外数据 RAM */
}
SECTIONS   /* 描述输入段如何被组合到输出段中，在存储器上如何放置输出段 */
{
     .vectors: {} > VECS          PAGE 0        /* 中断向量表 */
     .pvecs:   {} > PVECS         PAGE 0        /* 中断子向量表 */
     .text:         {} > PM           PAGE 0        /* 程序段 */
```

```
    .data:        {} > BLOCK_B0 PAGE 1        /* 数据段 */
}
```

4.4.3 DSP 的中断向量表和中断子向量表

本案例在介绍数字 PID 控制器和模糊 PI 控制器的 DSP 应用程序设计的同时，还介绍 TI 公司 C2000 系列 DSP 芯片开发时，中断向量表和中断子向量表的编写和配置方法。

一个实用 DSP 程序除包括主程序和系统初始化程序以及存储器配置文件之外，还需要有中断向量表和中断子向量表程序，对于一个 DSP 控制器来说，中断的使用和管理是不可缺少的。在系统中，控制器的作用就是控制整个系统实时、有序地按照程序的要求运行，而 DSP 只有一个 CPU，所以只有一个进程，当外部设备要求 DSP 控制时，就采用中断的方式，DSP 根据中断的优先级，通过响应中断并执行中断服务子程序（ISR）来对外部设备进行控制。一个正确的中断向量表和中断子向量表程序能够使系统正常运行，并能保证在系统不正常时自动恢复到程序初始化的状态，防止系统崩溃，提高工业控制的鲁棒性。

DSP LF2407 有两级中断，第一级中断是 CPU 中断，共 6 个；第二级中断是外围设备中断，共 46 个。由外设中断扩展控制器（PIE）和中断子向量表把外围设备中断映射到 CPU 中断，然后等待 CPU 的响应。此外 CPU 中断向量表还包括 19 个软件中断和硬件复位中断（Reset）以及一个不可屏蔽中断（NMI）。这种两级中断是采用集中化的中断扩展设计方法，特别适合有大量外设中断的工业控制系统。

以下就是采用通用定时器 GPT1 的比较操作来产生中断时，LF2407 的中断向量表和中断子向量表程序。当需要采用其他中断时，可以直接在此程序中根据需要修改。该程序对 LF2407 来说是通用的，只是发生中断时，CPU 要跳转的地址不一样而已。

```
;;;;;;;;;;;;;;;;;;;;    本程序的文件名：PID_generate_vec.asm   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        .include      "lf2407_regs.h"     ; 引用头部文件
                                          ; 建立中断向量表
        .sect     ".vectors"                    ; 定义主向量段，名称可以随意使用，一般采用.vectors
Reset_VECB    _cy_begin          ; 系统的复位中断向量，具有最高的优先级，程序地址是 0000h
INT1           B    PHANTOM      ;   CPU 的第一级中断向量，程序地址是 0002h
INT2           B    GISR2          ;   CPU 的第二级中断向量，程序地址是 0004h
INT3           B    PHANTOM      ;   CPU 的第三级中断向量，程序地址是 0006h
INT4           B    PHANTOM      ;   CPU 的第四级中断向量，程序地址是 0008h
INT5           B    PHANTOM      ;   CPU 的第五级中断向量，程序地址是 000Ah
INT6           B    PHANTOM      ;   CPU 的第六级中断向量，程序地址是 000Ch
RESERVED  B    PHANTOM   ; 程序空间保留，地址是 000Eh，但是为了保证中断向量表的
                                  ; 完整性，一定要在此处加入假中断向量 PHANTOM
                                  ; 以保证系统按照可控的方式运行
SW_INT8       B    PHANTOM          ; 软件中断向量 8，程序地址是 0010h
SW_INT9       B    PHANTOM          ; 软件中断向量 9，程序地址是 0012h
SW_INT10      B    PHANTOM          ; 软件中断向量 10，程序地址是 0014h
SW_INT11      B    PHANTOM          ; 软件中断向量 11，程序地址是 0016h
SW_INT12      B    PHANTOM          ; 软件中断向量 12，程序地址是 0018h
SW_INT13      B    PHANTOM          ; 软件中断向量 13，程序地址是 001Ah
SW_INT14      B    PHANTOM          ; 软件中断向量 14，程序地址是 001Ch
SW_INT15      B    PHANTOM          ; 软件中断向量 15，程序地址是 001Eh
SW_INT16      B    PHANTOM          ; 软件中断向量 16，程序地址是 0020h
TRAP      B    PHANTOM          ;   TRAP 指令中断，程序地址是 0022h
NMI           B    PHANTOM          ; 不可屏蔽中断，程序地址是 0024h
EMU_TRAP  B    PHANTOM          ; 程序空间保留，地址是 0026h
```

```
SW_INT20    B    PHANTOM          ; 软件中断向量 20，程序地址是 0028h
SW_INT21    B    PHANTOM          ; 软件中断向量 21，程序地址是 002Ah
SW_INT22    B    PHANTOM          ; 软件中断向量 22，程序地址是 002Ch
SW_INT23    B    PHANTOM          ; 软件中断向量 23，程序地址是 002Eh
SW_INT24    B    PHANTOM          ; 软件中断向量 24，程序地址是 0030h
SW_INT25    B    PHANTOM          ; 软件中断向量 25，程序地址是 0032h
SW_INT26    B    PHANTOM          ; 软件中断向量 26，程序地址是 0034h
SW_INT27    B    PHANTOM          ; 软件中断向量 27，程序地址是 0036h
SW_INT28    B    PHANTOM          ; 软件中断向量 28，程序地址是 0038h
SW_INT29    B    PHANTOM          ; 软件中断向量 29，程序地址是 003Ah
SW_INT30    B    PHANTOM          ; 软件中断向量 30，程序地址是 003Ch
SW_INT31    B    PHANTOM          ; 软件中断向量 31，程序地址是 003Eh

                                  ; 建立中断子向量表
            .sect    ".pvecs"             ; 定义中断子向量段
PVECTORS:
            B    PHANTOM  ; 假中断向量，中断子向量地址偏移量 0000h
            B    PHANTOM  ; 高优先级模式的外部引脚中断 1，中断子向量的偏移量 0001h
            B    PHANTOM  ; 假中断向量，中断子向量地址偏移量 0002h
            B    PHANTOM  ; 假中断向量，中断子向量地址偏移量 0003h
            B    PHANTOM  ; 高优先级模式的 ADC 中断，中断子向量地址偏移量 0004h
            B    PHANTOM  ; 高优先级模式的 SPI 中断，中断子向量地址偏移量 0005h
            B    PHANTOM  ; 高优先级模式的 SCI 接收中断，中断子向量地址偏移量 0006h
            B    PHANTOM  ; 高优先级模式的 SCI 发送中断，中断子向量地址偏移量 0007h
            B    PHANTOM  ; 假中断向量，中断子向量地址偏移量 0008h
            B    PHANTOM  ; 假中断向量，中断子向量地址偏移量 0009h
            B    PHANTOM  ; 假中断向量，中断子向量地址偏移量 000Ah
            B    PHANTOM  ; 假中断向量，中断子向量地址偏移量 000Bh
            B    PHANTOM  ; 假中断向量，中断子向量地址偏移量 000Ch
            B    PHANTOM  ; 假中断向量，中断子向量地址偏移量 000Dh
            B    PHANTOM  ; 假中断向量，中断子向量地址偏移量 000Eh
            B    PHANTOM  ; 假中断向量，中断子向量地址偏移量 000Fh
            B    PHANTOM  ; 假中断向量，中断子向量地址偏移量 0010h
            B    PHANTOM  ; 高优先级模式的外部引脚中断 2，中断子向量的偏移量 0011h
            B    PHANTOM  ; 假中断向量，中断子向量地址偏移量 0012h
            B    PHANTOM  ; 假中断向量，中断子向量地址偏移量 0013h
            B    PHANTOM  ; 假中断向量，中断子向量地址偏移量 0014h
            B    PHANTOM  ; 假中断向量，中断子向量地址偏移量 0015h
            B    PHANTOM  ; 假中断向量，中断子向量地址偏移量 0016h
            B    PHANTOM  ; 假中断向量，中断子向量地址偏移量 0017h
            B    PHANTOM  ; 假中断向量，中断子向量地址偏移量 0018h
            B    PHANTOM  ; 功率驱动保护中断 B，中断子向量地址偏移量 0019h
            B    PHANTOM  ; 假中断向量，中断子向量地址偏移量 001Ah
            B    PHANTOM  ; 假中断向量，中断子向量地址偏移量 001Bh
            B    PHANTOM  ; 假中断向量，中断子向量地址偏移量 001Ch
            B    PHANTOM  ; 假中断向量，中断子向量地址偏移量 001Dh
            B    PHANTOM  ; 假中断向量，中断子向量地址偏移量 001Eh
```

```
B     PHANTOM   ; 假中断向量，中断子向量地址偏移量 001Fh
B     PHANTOM   ; 功率驱动保护中断 A，中断子向量地址偏移量 0020h
B     PHANTOM   ; 比较器 1 中断，中断子向量地址偏移量 0021h
B     PHANTOM   ; 比较器 2 中断，中断子向量地址偏移量 0022h
B     PHANTOM   ; 比较器 3 中断，中断子向量地址偏移量 0023h
B     PHANTOM   ; 比较器 4 中断，中断子向量地址偏移量 0024h
B     PHANTOM   ; 比较器 5 中断，中断子向量地址偏移量 0025h
B     PHANTOM   ; 比较器 6 中断，中断子向量地址偏移量 0026h
B     PHANTOM   ; 定时器 1 周期中断，中断子向量地址偏移量 0027h
B     T1CINT_ISR     ; 定时器 1 比较中断，中断子向量地址偏移量 0028h
B     PHANTOM   ; 定时器 1 下溢中断，中断子向量地址偏移量 0029h
B     PHANTOM   ; 定时器 1 上溢中断，中断子向量地址偏移量 002Ah
B     PHANTOM   ; 定时器 2 周期中断，中断子向量地址偏移量 002Bh
B     PHANTOM   ; 定时器 2 比较中断，中断子向量地址偏移量 002Ch
B     PHANTOM   ; 定时器 2 下溢中断，中断子向量地址偏移量 002Dh
B     PHANTOM   ; 定时器 2 上溢中断，中断子向量地址偏移量 002Eh
B     T3GP_ISR       ; 定时器 3 周期中断，中断子向量地址偏移量 002Fh
B     PHANTOM   ; 定时器 3 比较中断，中断子向量地址偏移量 0030h
B     PHANTOM   ; 定时器 3 下溢中断，中断子向量地址偏移量 0031h
B     PHANTOM   ; 定时器 3 上溢中断，中断子向量地址偏移量 0032h
B     PHANTOM   ; 捕获器 1 中断，中断子向量地址偏移量 0033h
B     PHANTOM   ; 捕获器 2 中断，中断子向量地址偏移量 0034h
B     PHANTOM   ; 捕获器 3 中断，中断子向量地址偏移量 0035h
B     PHANTOM   ; 捕获器 4 中断，中断子向量地址偏移量 0036h
B     PHANTOM   ; 捕获器 5 中断，中断子向量地址偏移量 0037h
B     PHANTOM   ; 捕获器 6 中断，中断子向量地址偏移量 0038h
B     PHANTOM   ; 定时器 4 周期中断，中断子向量地址偏移量 0039h
B     PHANTOM   ; 定时器 4 比较中断，中断子向量地址偏移量 003Ah
B     PHANTOM   ; 定时器 4 下溢中断，中断子向量地址偏移量 003Bh
B     PHANTOM   ; 定时器 4 上溢中断，中断子向量地址偏移量 003Ch
B     PHANTOM   ; 假中断向量，中断子向量地址偏移量 003Dh
B     PHANTOM   ; 假中断向量，中断子向量地址偏移量 003Eh
B     PHANTOM   ; 假中断向量，中断子向量地址偏移量 003Fh
B     PHANTOM   ; 高优先级模式的 CAN 邮箱中断，中断子向量地址偏移量 0040h
B     PHANTOM   ; 高优先级模式的 CAN 错误中断，中断子向量地址偏移量 0041h
```

对于一个实际的 DSP 系统来说，系统的中断管理是不可缺少的，因为目前任何 DSP 实时系统都具有中断，中断是 DSP 系统和外部世界发生实时联系的一个重要手段。作为 TI 公司的 C2000 系列 DSP，它是偏向于控制的 DSP 芯片，因此它的中断管理更丰富和先进，读者理解起来也较困难。LF2407 支持 6 个一级可屏蔽中断，采用集中化的中断扩展设计来满足大量的外设中断请求，所以每一级中断又有多个中断源，例如一级中断 INT2 包含的中断源有比较器 1、2、3、4、5、6 中断，定时器 1、3 的周期、比较、下溢、上溢中断等。为了正确地响应外设中断，应该分两步来完成中断服务子程序。在本案例的程序中，采用通用定时器 1 的比较匹配中断，当 CPU 响应该中断时，首先转移到一级中断 INT2 中，也即 GISR2 处执行，在一级中断 INT2 中读取外设中断向量寄存器（PIVR）的值，它是个偏移量，再加上中断子向量的首地址，程序就可以转移到二级中断子向量 T1CINT_ISR（定时器 1 比较中断）子程序中，执行相应的操作后即完成了一次中断调用。

LF2407 通过外设中断扩展控制器（PIE）来实现集中化的中断扩展管理，这可以实现在占用极少资源的情况下，大大扩展可用的中断源。因此在实际的 DSP 程序中，中断向量表和中断子向量表程序是不可缺

少的，读者在自行编写程序时一定要在主程序中用.include 汇编伪指令把本实例介绍的向量表文件 PID_generate_vec.asm 包括进来。

```
.include  "pwm_generate_vec.asm"   ; 引用中断向量表和中断子向量表程序
```

另外，假中断向量是 LF2407 的一个特有的概念，它是保持中断系统完整性的一个特性。在向量表文件 PID_generate_vec.asm 中可以发现，中断子向量表的地址是从 00h 到 42h，而中断源却没有那么多，并且其入口地址又是固定的，所以在中断子向量表的没有中断源的地方就要插入假中断向量以保证中断系统的完整性。当一个中断已经被响应，但却没有外设将中断向量的地址偏移量装入中断向量寄存器（PIVR）中时，假中断向量的地址就被装入 PIVR 中，这种缺省保证了系统按照可以控制的方式进行处理。假中断向量的子程序如下所示，一般都是复位看门狗电路。

```
        B     PHANTOM       ; 跳转到假中断向量子程序中
PHANTOM :                   ; 假中断向量子程序
        LDP #DP_PF1
        SPLK #05555h,  WDKEY        ; 复位看门狗
        SPLK #0AAAAh,  WDKEY
        RET
```

4.4.4 数字 PID 控制器的 DSP 应用程序

下面介绍包含数字 PID 控制器算法的 DSP 应用程序，该程序是基于图 4.2 的硬件平台和图 4.6 的原理框图实现数字 PID 控制器的，程序的文件名是“Digital_PID.asm”，程序中具体的变量定义如表 4.1 所示。在下面的程序代码中都添加了详细的解释说明。

```
        ; 实用数字 PID 控制程序
        .title   "Digital_PID.asm"       ; 汇编伪指令，定义程序的文件名
        .include      "lf2407_regs.h"              ; 引用头部文件
        .include      "Digital_PID_vec.asm"   ; 引用中断向量表和中断子向量表
        .def          _cy_begin                ; 定义程序的入口地址

Kp_init          .set  100h              ; 比例增益系数 Kp 的初始值
Ki_high_init      .set  00h                  ; 积分增益系数 Ki 的高位初始值
Ki_low_init      .set  1000h                 ; 积分增益系数 Ki 的低位初始值
Kd_init          .set  05h               ; 微分增益系数 Kd 的初始值
PID_ref_init      .set  500              ; 数字 PID 控制器的参考输入初始值
                              ; 本程序实现电机速度 PID 控制，设置参考输入是 500 转/分钟
PID_output_MAX   .set  07FFFh            ; 数字 PID 控制器输出控制量最大值限幅
PID_output_MIN   .set  0h                ; 数字 PID 控制器输出控制量最小值限幅

       .data                             ; 定义数据区中的变量
ADRESULT      .word  00h             ; 存储当前 AD 转换的结果
PID_input       .word  00h               ; 存储变量 ADRESULT，也即当前实际输入量 c（k）
PID_output      .word  00h               ; 存储数字 PID 控制器的当前控制量 u（k）
PID_output1     .word  00h               ; 存储数字 PID 控制器的上次控制量 u（k—1）
PID_reference   .word  00h               ; 存储数字 PID 控制器的当前参考输入量 r（k）
PID_e0          .word  00h           ; 存储数字 PID 控制器的当前偏差量 e（k）=r（k）—c（k）
PID_e1          .word  00h               ; 存储数字 PID 控制器的上次偏差量 e（k—1）
PID_e2          .word  00h               ; 存储数字 PID 控制器的上上次偏差量 e（k—2）
```

```
Kp              .word   00h                     ; 存储比例增益系数 Kp
Ki_high         .word   00h                     ; 存储积分增益系数 Ki 的高位字
Ki_low          .word   00h                     ; 存储积分增益系数 Ki 的低位字
Kd              .word   00h                     ; 存储微分增益系数 Kd
A_coeff_high    .word   00h                     ; 存储 A 系数高位
A_coeff_low     .word   00h                     ; 存储 A 系数低位
B_coeff         .word   00h                     ; 存储 B 系数

tmp1_high       .word   00h                     ; 暂存单元 1 的高位字
tmp1_low        .word   00h                     ; 暂存单元 1 的低位字
tmp2_high       .word   00h                     ; 暂存单元 2 的高位字
tmp2_low        .word   00h                     ; 暂存单元 2 的低位字
tmp3            .word   00h                     ; 暂存单元 3
e0_sign         .word   00h                     ; 用来存储当前偏差量 e（k）的符号
abs_e0          .word   00h                     ; 用来存储当前偏差量 e（k）的绝对值

          .text                                 ; 以下是程序段
_cy_begin :                                     ; 程序的入口地址
          NOP
          CALL   system_init                    ; 调用系统初始化子程序
          CALL   PID_init                       ; 调用数字 PID 控制的初始化子程序
          CALL   cy_AD_init                     ; 调用 DSP 片上 AD 转换器的初始化子程序
cy_LOOP   :                   ; 主程序进入循环等待，程序在高优先级模式的 ADC 中断控制下运行
          NOP                       ;   ADC 中断一次，采样一个实际输出值，并进行一次 PID 控制
          NOP
          B       cy_LOOP

system_init :                                   ; 系统初始化子程序
          SETC   INTM                   ; 关闭系统的全局中断
          CLRC   OVM                    ; 置系统溢出模式为 0，表示累加器中的结果正常溢出
          CLRC   SXM                    ; 置符号扩展方式为 0，表示系统将抑制符号扩展
          CLRC   CNF                    ; 使 CNF=0，DARAM 被映射到数据存储空间 B0 区
          LDP    #DP_PF1                ; 指向 7000h～707Fh 外设寄存器区
          SPLK   #081FEh, SCSR1             ;   CLKOUT 引脚输出 CPU 时钟，  PLL 的倍频系数是 4
                                        ; 也即 CLKIN=6 MHz，而 CLKOUT=24 MHz
          SPLK   #0E8h, WDCR            ; 禁止 WDT
          LDP    #0                     ; 指向数据页的第 0 区
          SPLK   #0001h, IMR            ; 使能中断第 1 级 INT1
          SPLK   #0FFFFh, IFR           ; 清全部一级中断标志
          RET

cy_AD_init :                                        ;   AD 转换模块初始化子程序
          LDP     #DP_EVB               ; 指向 7500h～757Fh 外设 EVB 寄存器区
          SPLK   #0000h, T4CNT              ; 定时器 4 的计数寄存器清 0
          SPLK   #0176h, T4PR           ; 定时器 4 的周期寄存器设置为 374，表示周期计数是 1ms
          SPLK   #0400h, GPTCONB            ; 位[10－9]（T4TOADC）=10，表示由通用定时器的
                                        ; 周期中断标志来启动模数转换
```

```
                                        ; 位 6（TCOMPOE）=0，禁止所有定时器的比较输出
        SPLK  #0160Ch, T4CON          ; 配置定时器 4 的控制寄存器
                                ; 位[12－11]，TMODE[1－0]=10，选择连续增计数模式
                                ; 位[10－8]，TPS[2－0]=110，选择输入时钟预定标系数=64
                                ; 位 7，T4SWT3=0，表示定时器 4 使用自身的周期寄存器 T4PR
                            ; 位 6，TENABLE=0，先禁止定时器的操作，等配置完后再打开
                            ; 位[5－4]，TCLKS[1－0]=00，选择内部 CPU 时钟
                            ; 位[3－2]，TCLD[1－0]=11，定时器比较器的重装载条件，现在保留
                            ; 位 1，TECMPR =0，禁止定时器的比较操作
                                        ; 位 0，SELT3PR=0，周期寄存器选择位
        SPLK  #0FFFFh, EVBIFRA          ; 清 EVB 模块的全部中断标志寄存器
        SPLK  #0FFFFh, EVBIFRB
        SPLK  #0FFFFh, EVBIFRC
        SPLK  #00000h, EVBIMRA
        SPLK  #00001h, EVBIMRB          ; 仅仅使能 EVB 模块的定时器 4 的周期中断 T4PINT
        SPLK  #00000h, EVBIMRC
        LDP   #DP_PF2                   ; 指向 7080h～70FFh 外设寄存器区
        SPLK  #0000h, ADCCTRL1          ; 配置 ADC 控制寄存器 1
                    ; 位[11－8]=0000，表示 ADC 的采样时间是 2×CPU 时钟周期
                    ; 位 7=0，是 CPS 位，表示 ADC 的转换时钟预定标就是 CPU 时钟频率
                                        ; 位 6=0，是连续转换位，现在采用启动/停止模式
                                        ; 位 5=0，表示 ADC 转换的中断请求是高优先级的
                                        ; 位 4=0，表示 ADC 转换器采用双排序器工作模式
                                        ; 位 0=0，禁止自测试使能模式
        SPLK  #8404h, ADCCTRL2          ; 配置 ADC 控制寄存器 2
                                        ; 位 15=1，表示不使能 ADC 的级连排序器
                ; 位[11－10]=01，表示 ADC 的排序器 1 在中断标志位 1 置 1 时立即申请中断
                    ; 位 9=0，表示目前 ADC 的排序器 1 无中断事件发生
                    ; 位[3－2]=01，表示 ADC 的排序器 2 在中断标志位 2 置 1 时立即申请中断
                                        ; 位 0=0，ADC 的排序器 2 不能被 EVB 触发源启动
        SPLK  #0001h, MAXCONV       ; 程序中只有一路 AD 采样，所以最大转换通道为 1
        SPLK  #0000h, CHSELSEQ1         ; 配置输入通道选择排序控制寄存器
        SPLK  #0000h, CHSELSEQ2     ; 目前程序中只有一路 AD 采样，所以只用到了一个通道
        SPLK  #0000h, CHSELSEQ3         ; 用到的通道号是 0 号
        SPLK  #0000h, CHSELSEQ4
        CALL   AD_Start                 ; 配置完所有寄存器后，启动 AD 转换信号
        LDP   #4                        ; 指向 200h～27Fh 数据存储器的 B0 区
        SPLK  #0000h,  ADRESULT ; 初始化用户变量 ADRESULT，程序运行时存 AD 转换结果
        RET
AD_Start  :                             ; 启动 AD 转换信号
        LDP   #DP_EVB
        LACL  T4CON                 ; 把定时器 4 的控制寄存器位 6（TENABLE）置 1
        OR    #0040h                    ; 使能定时器的操作
        SACL  T4CON
        RET

PID_Control :                           ; 根据输入值进行 PID 算法的计算
```

```
SETC  SXM                          ；置符号扩展方式为1，表示系统进行符号扩展
SETC  OVM                      ；置系统溢出模式为1，溢出时使用最大正值和负值
SPM   #0                           ；表示乘法器不移位直接装入CALU
LDP   #4                           ；指向用户变量区200h～27Fh
LACL  ADRESULT                     ；把本次AD转换的结果存储到变量PID_input中
SACL  PID_input
LACC  Ki_high, 16                  ；把积分增益系数 Ki的高位字左移16位加到ACC中
ADDS  Ki_low               ；无符号加法，表示把 Ki的高位字和低位字拼接起来加载到ACC
ADD   Kp, 16                       ；  ACC=Kp+Ki
ADD   Kd, 16                       ；  ACC=Kp+Ki+Kd
SACH  A_coeff_high                 ；系数 A高位字=高位（Kp+Ki+Kd）
SACL  A_coeff_low                  ；系数 A低位字=低位（Kp+Ki+Kd）
LACC  Kd, 16
SFL                                ；此时，ACC=2×Kd
ADD   Kp, 16                       ；  ACC=2×Kd+Kp
SACH  B_coeff                      ；系数 B = 2×Kd+Kp
LACC  PID_reference                ；加载输入参考值
SUB   PID_input                    ；  ACC=PID_reference - PID_input
SACL  PID_e0                       ；本次输入的偏差量，e(k)= r(k) - c(k)
LT    Kd
MPY   PID_e2                       ；  Kd× e(k－2)
PAC                                ；把乘积寄存器移位后加载到累加器，此处移位值是0
LT    B_coeff                      ；把 B系数加载到T寄存器中
MPY   PID_e1                       ；  B× e(k－1)=(2×Kp+Ki)× e(k－1)
SPAC                           ；此时ACC=Kd× e(k－2) - (2×Kp+Ki)× e(k－1)
SACH  tmp1_high, 1                 ；把ACC的高位字放到暂存字 tmp1_high中
SACL  tmp1_low,  1                 ；把ACC的低位字放到暂存字 tmp1_low中
                                   ；此处开始求最后一项增量，也即当前增量
LACC  PID_e0                       ；加载当前偏差量 e(k)到累加器ACC中
SACL  e0_sign                  ；把 e(k)存储到变量 e0_sign中，便于后面进行符号判断
ABS                                ；对 e(k)取绝对值，为无符号乘法做准备
SACL  abs_e0
LT    abs_e0                       ；把| e(k)| 加载到T寄存器中
MPYU  A_coeff_low                  ；无符号乘法，A低位字×| e(k)|
SPH   tmp2_low                     ；把乘积寄存器PREG的高16位存储到变量 tmp2_low
MPYU  A_coeff_high                 ；无符号乘法，A高位字×| e(k)|
PAC                                ；把 A高位字×| e(k)| 的乘积加载到累加器
ADDS  tmp2_low                 ；  ACC=A高位字×| e(k)|+A低位字×| e(k)|
SACH  tmp2_high, 1                 ；把该结果存储到暂存器变量 tmp2_high和 tmp2_low
SACL  tmp2_low,  1
LACC  e0_sign                      ；判断当前偏差量 e(k)的符号
BCND  cy_DONE,  GT                 ；当 e(k)＞0时，不必再进行处理，跳转
LACC  tmp2_high, 16                ；当 e(k)<= 0时
ADDS  tmp2_low       ；使ACC=A系数高位字×| e(k)|+A系数低位字×| e(k)|
NEG                                ；因为 e(k)<= 0，所以使ACC取负值
SACH  tmp2_high          ；把取负后的ACC存储到暂存器变量 tmp2_high和 tmp2_low
SACL  tmp2_low
```

```
cy_DONE :                    ; 对当前偏差量 e(k)的符号判断处理结束
        LACC  tmp1_high, 16    ; 加载[Kp×e(k-2) - (2×Kp+Ki)×e(k-1)]的高位字
        ADDS  tmp1_low                ;  ACC=Kp×e(k-2) - (2×Kp+Ki)×e(k-1)
        ADDS  tmp2_low
        ADD   tmp2_high, 16  ; 此时 ACC 存放数字 PID 控制器的控制增量△u(k)=u(k) - u(k-1)
        SACH  tmp3   ;  tmp3=△u(k)=A×e(k) - (2×Kp+Ki)×e(k-1)+Kp×e(k-2)
        LACC  PID_output1, 16          ; 加载上次控制量 u(k-1)到 ACC，并左移 16 位
        ADD   tmp3, 16                 ; 得到本次的控制量，存放在 ACC 中，字长是 32 位
        SACH  PID_output ; 取 ACC 的高 16 位存放到本次控制量的变量中，16 位已经能满足精度
              ;  u(k)=u(k-1)+A×e(k) - (2×Kp+Ki)×e(k-1)+Kp×e(k-2)
        LACC  PID_output            ; 变量 PID_output 中存放送到 DA 的控制量
        SUB   #PID_output_MAX       ; 对输出控制量进行限幅控制
        BCND  greater_MAX, GT       ; 进行上限幅
        LACC  PID_output
        SUB   #PID_output_MIN
        BCND  less_MIN, LT          ; 进行下限幅
        B     PID_end
greater_MAX  :          ; 当 PID_output > PID_output_MAX 时，令 PID_output=PID_output_MAX
        SPLK  #PID_output_MAX, PID_output
        B     PID_end
less_MIN :              ; 当 PID_output < PID_output_MIN 时，令 PID_output=PID_output_MIN
        SPLK  #PID_output_MIN, PID_output
PID_end :                           ; 为下一次 PID 控制做准备
        LDP   #4                    ; 指向用户变量区 200h~27Fh
        LACC  PID_e1
        SACL  PID_e2                ; 使得 e(k-2)=e(k-1)
        LACC  PID_e0
        SACL  PID_e1                ; 使得 e(k-1)=e(k)
        LACC  PID_output
        SACL  PID_output1               ; 使得 PID_output1=PID_output
        CLRC  SXM                   ;  SXM=0，抑制符号位扩展
        RET
PID_init :                          ; 数字 PID 控制器输入值和参数初始化
        LDP   #4                    ; 指向用户变量区，200h~27Fh 的数据存储器的 B0 区
        SPLK  #Kp_init, Kp              ; 初始化比例增益系数 Kp
        SPLK  #Ki_high_init, Ki_high    ; 初始化积分增益系数高位字 Ki_high
        SPLK  #Ki_low_init, Ki_low      ; 初始化积分增益系数低位字 Ki_low
        SPLK  #Kd_init, Kd              ; 初始化微分增益系数 Kd
        SPLK  #PID_ref_init, PID_reference   ; 初始化系统参考输入值 PID_reference=r(t)
        SPLK  #0, PID_e2                ; 初始化上上次的偏差量 e(k-2)
        SPLK  #0, PID_e1                ; 初始化上次的偏差量 e(k-1)
        SPLK  #0, PID_e0                ; 初始化当前的偏差量 e(k)
        SPLK  #0, PID_output1            ; 初始化上次的控制量 u(k-1)
        SPLK  #0, PID_output             ; 初始化当前控制量 u(k)
        RET
GISR1 :                             ; 优先级 INT1 中断入口
                                    ; 保护现场
```

```
        LDP     #DP_PF1                 ;  DP 指针指向 7000h~707Fh 外设数据区
        LACC PIVR, 1                ; 读取外设中断向量寄存器(PIVR)，并左移一位
        ADD     #PVECTORS           ; 再加上外设中断子向量的入口地址
        BACC                            ; 根据累加器的值，跳到相应的中断服务子程序中
                                        ; 此处是高优先级的 ADC 中断
ADCINT_ISR :                            ; 高优先级模式的 ADC 中断子程序
        CLRC   SXM                  ; 抑制符号位扩展
        LDP    #4                       ; 指向用户变量区
        LAR    AR2, #RESULT0            ; 把当前 AD 转换的结果存储到 AR2 中
        MAR    *, AR2                   ; 把 AR2 设置为当前辅助寄存器
        LACC   *, 10  ; 由于 DSP 内嵌的 AD 转换器是 10 位的，所以左移 10 位加载到 ACC 高位
        SACH   ADRESULT             ; 变量 ADRESULT 存储当前 AD 转换的结果
        CALL   PID_Control              ; 调用数字 PID 控制子程序
        CLRC   INTM                 ; 开总中断，因为一进中断就自动关闭总中断
        RET                             ; 中断返回

PHANTOM :                               ; 假中断向量子程序
        LDP  #DP_PF1                ;  DP 指针指向 7000h~707Fh 外设数据区
        SPLK  #05555h, WDKEY            ; 复位看门狗
        SPLK  #0AAAAh, WDKEY
        RET
        .end                            ; 程序结束
```

4.4.5 模糊 PI 控制器的 DSP 应用程序

下面介绍包含模糊 PI 控制器算法的 DSP 应用程序，该程序是基于图 4.2 的硬件平台和图 4.8 的原理框图实现模糊 PI 控制器的，下面的程序段就是数字模糊 PI 控制器的控制表程序，程序的文件名是 Fuzzy_PI.asm，每行程序都有详细的解释说明，由于模糊 PI 控制器是在数字 PID 控制器的基础上改进的，又由于篇幅的限制，本程序没有给出所有的程序，其主程序和中断子程序以及初始化程序和数字 PID 控制器的程序是基本相同的，在此不再赘述。

```
; 实用模糊 PI 控制程序
PI_Control :                ; 此处开始执行 PI 控制器
        SETC  SXM
        SETC  OVM
        SPM   #0
        LDP   #4
        LACL  ADRESULT              ; 把 AD 采样的结果存储到变量 PID_input
        SACL  PID_input
        CALL  Fuzzy_PI_table            ; 此处调用模糊 PI 控制规则
                                        ; 以下程序实现模糊 PI 控制规则表，如表 4.3 所示
Fuzzy_PI_table :
        LACC  PID_reference             ; 加载系统参考输入量 r(k)
        SUB   PID_input
        SACL  PID_e0                    ; 求得系统当前偏差量 e(k)
        SUB   PID_e1                    ; 与上次系统偏差量相减
        SACL  PI_delta_e            ; 求得系统当前偏差变化率△e(k)=e(k)-e(k-1)
        LT    q1                        ; 对偏差量进行模糊化，q1 是量化因子
```

```
        MPY   PID_e0
        PAC                              ; 求出模糊偏差量=q1×e(k)
        SACL  Fuzzy_PI_e                 ; 把模糊偏差量存储到变量 Fuzzy_PI_e 中
        LT    q2                         ; 对偏差量变化率进行模糊化，q2 是量化因子
        MPY   PI_delta_e                 ; 模糊偏差变化率=q2×△e(k)
        PAC
        SACL  Fuzzy_PI_delta_e           ; 把模糊偏差变化率存储到变量 Fuzzy_PI_delta_e 中
        LACL  Fuzzy_PI_e                 ; 对模糊控制规则进行限幅控制
        SUB   #6
        BCND  Normal_Fuzzy_PI_1,  LT     ; 当模糊偏差量大于等于 6 时
        SPLK  #6,  Fuzzy_PI_e            ; 模糊偏差量取最大值 6
Normal_Fuzzy_PI_1 :
        LACL  Fuzzy_PI_delta_e
        SUB   #6
        BCND  Normal_Fuzzy_PI_2,  LT     ; 当模糊偏差变化率大于等于 6 时
        SPLK  #6,  Fuzzy_PI_delta_e      ; 模糊偏差变化率取最大值 6
Normal_Fuzzy_PI_2 :                      ; 以下程序是模糊控制规则求解
       LACL  Fuzzy_PI_e
       SUB   #5                          ; 当模糊偏差量=5 时
       BCND  Fuzzy_PID_e_5,  EQ
         B     Fuzzy_PID_e_not5               ; 如果不相等，则跳出该条模糊控制规则
Fuzzy_PID_e_5 :                                 ; 继续寻找合适的控制规则
         LACL  Fuzzy_PI_delta_e                 ; 当模糊偏差变化率=1 时
       SUB   #1
       BCND  Fuzzy_PID_delta_e_1,  EQ
         B     Fuzzy_PID_e_not5          ; 如果不相等，则跳出该条模糊控制规则
Fuzzy_PID_delta_e_5 :
       SPLK  #5,  Fuzzy_KP               ; 这时，模糊比例增益系数 KP=5
       SPLK  #0,  Fuzzy_KI               ; 模糊比例增益系数 KI=0
       B Fuzzy_PI_end                ; 此时已经找到一条符合要求的模糊控制规则
                                         ; 跳出模糊规则求解程序
Fuzzy_PID_e_not5:
       LACL  Fuzzy_PI_e                  ; 当模糊偏差量=4 时
       SUB   #4
       BCND  Fuzzy_PID_e_4, EQ
        B     Fuzzy_PID_e_not4           ; 如果不相等，则跳出该条模糊控制规则
Fuzzy_PID_e_4 :
      LACL  Fuzzy_PI_delta_e             ; 当模糊偏差变化率=2 时
       SUB   #2
       BCND  Fuzzy_PID_delta_e_2,  EQ
      B     Fuzzy_PID_e_not4             ; 如果不相等，则跳出该条模糊控制规则
Fuzzy_PID_delta_e_5 :
       SPLK  #4,  Fuzzy_KP               ; 这时，模糊比例增益系数 KP=4
       SPLK  #1,  Fuzzy_KI               ; 模糊比例增益系数 KI=1
       B Fuzzy_PI_end                ; 此时已经找到一条符合要求的模糊控制规则
Fuzzy_PID_e_not4 :
       LACL  Fuzzy_PI_e                  ; 当模糊偏差量=3 时
```

```
        SUB   #3
        BCND  Fuzzy_PID_e_3, EQ          ；如果相等，则继续向下判断
         B     Fuzzy_PID_e_not3          ；如果不相等，则跳出该条模糊控制规则
Fuzzy_PID_e_3 :
        LACL  Fuzzy_PI_delta_e           ；当模糊偏差变化率=3 时
        SUB   #3
        BCND  Fuzzy_PID_delta_e_3,  EQ
         B     Fuzzy_PID_e_not3          ；如果不相等，则跳出该条模糊控制规则
Fuzzy_PID_delta_e_3 :
        SPLK  #3,   Fuzzy_KP             ；这时，模糊比例增益系数 KP=3
        SPLK  #2,   Fuzzy_KI             ；模糊比例增益系数 KI=2
        B  Fuzzy_PI_end                  ；此时已经找到一条符合要求的模糊控制规则
                                         ；跳出模糊规则求解程序
Fuzzy_PID_e_not3 :
……  ……  ……  ……  ……     ；和上面的程序段一样描述出表 4.3 所示的所有模糊控制规则
……  ……  ……  ……  ……     ；此处省略了编程结构相同的模糊控制规则
Fuzzy_PI_end :                   ；此处模糊控制规则求解结束，对模糊输出控制里进行反模糊化
        LT     k1                        ；对模糊比例增益系数 KP 反模糊化
        MPY    Fuzzy_KP                  ；  k1×Fuzzy_KP
        PAC
        SACL   Kp                        ；求得反模糊化结果 Kp=k1×KP
        LT     k2                        ；对模糊积分增益系数 KI 反模糊化
        MPY    Fuzzy_KI                  ；  k2×Fuzzy_KI
        PAC
        SACL   Ki                        ；求得反模糊化结果 Ki=k2×KI
        RET                              ；模糊控制规则求解程序结束，返回调用处
```

4.5 控制系统的性能评估

本节将对系统的开环特性，以及加上控制算法后的控制性能作对比，由此反映数字 PID 控制和模糊 PI 控制算法的特点。

4.5.1 系统的开环特性

图 4.12、图 4.13 和图 4.14 给出了伺服电机的动态速度特性。图中所示是在输入端分别加上 500～1000mV、1500～2000mV、2500～3000mV 电压的阶跃信号后电机的速度响应。曲线 a 为常温下空载速度阶跃响应，b 为高温（80℃左右）下空载速度阶跃响应，c 为负载 3Kgf·cm 的速度阶跃响应。在温度和所加负载发生变化时电机的速度特性也随着变化，且速度越高变化越大。

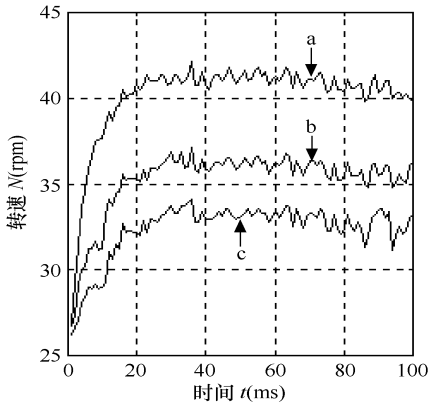


图 4.12　500～1000mV 的速度阶跃响应

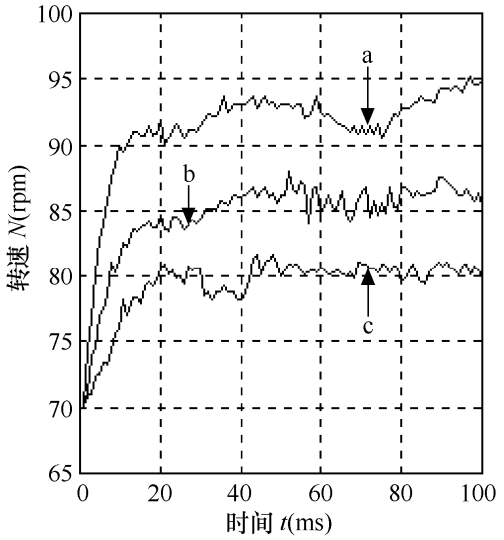


图 4.13　1500～2000mV 的速度阶跃响应

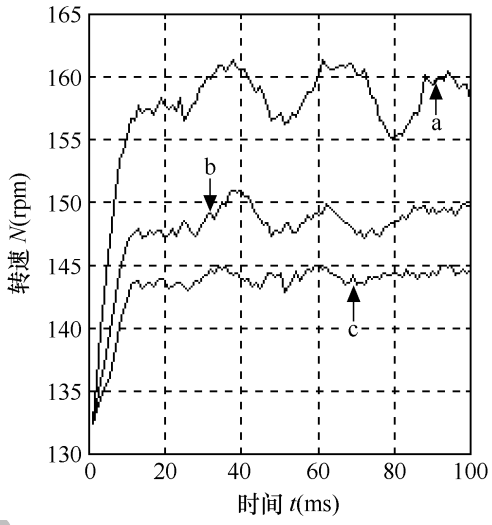
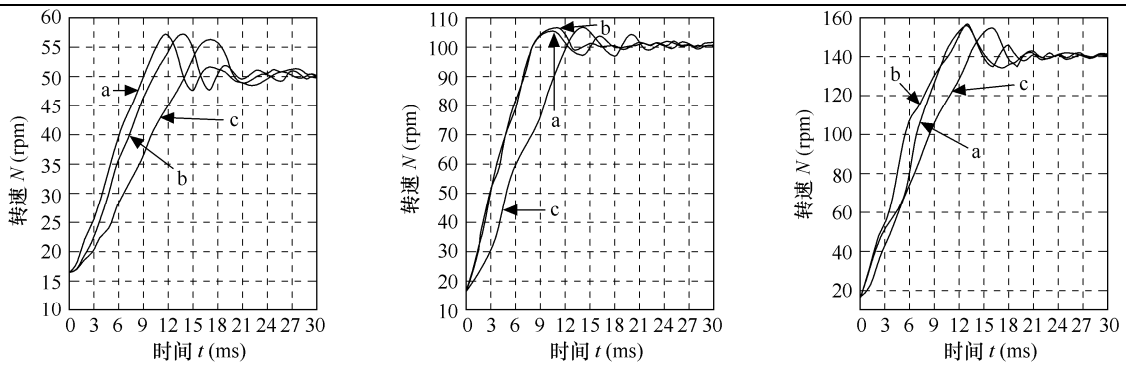


图 4.14　2500～3000mV 的速度阶跃响应

4.5.2　数字 PID 控制特性

图 4.15 给出了在数字 PID 控制下伺服电机的动态速度特性。设计数字 PID 控制器后，做了一系列实验。图 4.15 是电机在数字 PID 控制下不同设定速度的阶跃响应。其中，曲线 a 表示电机空载情况下的阶跃响应；曲线 b 表示电机长时间运行后温度上升的情况下的阶跃响应；曲线 c 表示当电机加上 3Kgf·cm 的负载（60%的额定负载）时，电机的阶跃响应曲线。由图 4.15 可以看出，只有在电机的速度设定为 100rpm 时，数字 PID 控制器的效果较好，不管在哪种情况下，最大超调量均小于 7%，而在 50rpm 和 140rpm 的设定速度下，最大超调量均大于 10%。另外在设定速度为 100rpm 时，电机阶跃响应的上升时间和调整时间均比其他的设定速度下电机阶跃响应的上升时间和调整时间小。但是随着电机温度上升或加上负载，PID 控制器的控制品质相对下降，超调量增大同时调整时间变长。这是因为数字 PID 控制器的参数是在电机空载，设定速度为 100rpm 的情况下调整的，所以在这种情况下 PID 控制器的控制品质相对较好。而当电机设定速度变化或电机带负载运行或温度变化时，由于电机的特性随电机带载情况或电机温度的变化而改变，因此电机的工作点发生漂移，而 PID 控制器的参数无法跟着改变，从而控制品质相对下降。



(1) 设定速度 50rpm　　(2) 设定速度 100rpm　　(3) 设定速度 140rpm

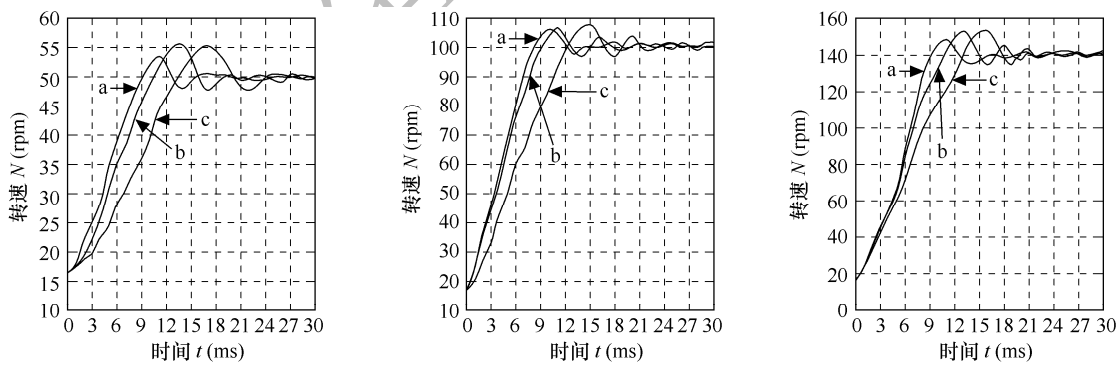
说明：曲线 a 空载情况，曲线 b 长时间运行后温度上升的情况，曲线 c 加上 3Kgf • cm 的负载

图 4.15　在数字 PID 控制下不同设定速度的阶跃响应

电机在 PID 控制下无法完全消除静差，这是因为电机本身的时间常数小，动态响应快，惯性小，容易引入干扰，而 PID 控制中的微分环节（D）虽然能改善系统的动态特性，加快系统响应，降低超调，但是它也使系统的抗干扰能力下降，所以电机在 PID 控制下进入稳态后，由于外界干扰的存在，使得电机总是存在略微的抖动。

4.5.3　模糊 PI 控制特性

设计模糊 PI 控制器后，做了一系列实验。图 4.16 是电机在模糊 PI 控制下，不同设定速度的阶跃响应。其中，曲线 a 表示电机空载情况下的阶跃响应；曲线 b 表示电机长时间运行后温度上升的情况下的阶跃响应；曲线 c 表示当电机加上 3Kgf • cm 的负载时，电机的阶跃响应曲线。由图 4.16 可以看出，电机在常温空载情况下，不管设定速度是多少，参数自整定模糊 PI 控制器的控制效果都比较好，最大超调量均小于 7%，而调整时间均在 16ms 以内，并且进入稳态后速度抖动较小。但是随着电机温度上升或所加负载发生变化时，控制器的控制品质相对下降，超调量增大同时调整时间变长。这是因为模糊 PI 控制器的参数 K_p 和 K_i 是在电机空载，设定速度为 100rpm 的情况下给出的，同时用于调节 PI 参数 K_p 和 K_i 的模糊控制规则是在常温下，电机空载时给出的，所以在电机空载情况下控制器的控制品质相对较好。



(1) 设定速度 50rpm　　(2) 设定速度 100rpm　　(3) 设定速度 140rpm

说明：曲线 a 空载情况，曲线 b 长时间运行后温度上升的情况，曲线 c 加上 3Kgf • cm 的负载

图 4.16　在模糊 PI 控制下不同设定速度的阶跃响应

联系方式

集团官网：www.hqyj.com　　嵌入式学院：www.embedu.org　　移动互联网学院：www.3g-edu.org

企业学院：www.farsight.com.cn　　物联网学院：www.topsight.cn　　研发中心：dev.hqyj.com

集团总部地址：北京市海淀区西三旗悦秀路北京明园大学校内 华清远见教育集团

北京地址：北京市海淀区西三旗悦秀路北京明园大学校区，电话：010-82600386/5

上海地址：上海市徐汇区漕溪路银海大厦 A 座 8 层，电话：021-54485127

深圳地址：深圳市龙华新区人民北路美丽 AAA 大厦 15 层，电话：0755-22193762

成都地址：成都市武侯区科华北路 99 号科华大厦 6 层，电话：028-85405115

南京地址：南京市白下区汉中路 185 号鸿运大厦 10 层，电话：025-86551900

武汉地址：武汉市工程大学卓刀泉校区科技孵化器大楼 8 层，电话：027-87804688

西安地址：西安市高新区高新一路 12 号创业大厦 D3 楼 5 层，电话：029-68785218