DSP
控制技术与应用

DSP KONGZHI JISHU YU YINGYONG

任志斌　杨　勇　编著

中国电力出版社
CHINA ELECTRIC POWER PRESS

本学术著作获江西理工大学优秀学术著作出版基金资助

DSP
控制技术与应用

DSP KONGZHI JISHU YU YINGYONG

任志斌　杨　勇　编著

中国电力出版社
CHINA ELECTRIC POWER PRESS

内 容 提 要

本书首先介绍了TI公司TMS320F2812芯片编程的基础知识，然后以实现电动机控制及新能源逆变控制技术为重点，介绍了TI公司的TMS320F2812芯片在电动机控制系统及新能源光伏发电中的应用。全书共6章，第1章介绍了TMS320F2812的主要原理，包括DSP的结构及性能、存储空间及时钟、中断系统、事件管理器和A/D转换器；第2章围绕DSP控制技术方面的编程需要，重点介绍了定点DSP的数据Q格式、带有死区的PWM波形、数字PI调节器的DSP实现方法、数字测速及电压空间矢量PWM（SVPWM）控制技术的DSP实现方法；第3章针对控制系统的实现，介绍了硬件电路，包括辅助电源电路、功率电路和控制电路；第4章至第6章以DSP的控制在实际项目应用中的实例分别对无刷直流电动机控制器、永磁同步电动机通用伺服控制器和光伏发电逆变控制器的DSP控制技术作了详细介绍。

本书以DSP的电气控制热点应用项目为重点，原理分析通俗易懂，各个环节都有作者在实践中的实例，通过实例使读者加深对内容的理解，全书的讲解通俗易懂、深入浅出。本书适合作为电气工程及其自动化、自动化、电动机与电器、电力电子与电力传动专业及其他相关专业教材，也可以作为工程技术人员研究、开发电气控制系统的参考书。

图书在版编目（CIP）数据

DSP控制技术与应用/任志斌，杨勇编著. —北京：中国电力出版社，2015.5

ISBN 978-7-5123-7[illegible]-9

Ⅰ. ①D… Ⅱ. ①任… ②杨… Ⅲ. ①数字信号处理—应用—电机—控制系统 Ⅳ. ①TM301.2

中国版本图书馆CIP数据核字（2015）第045558号

中国电力出版社出版、发行

（北京市东城区北京站西街19号 100005 http://www.cepp.sgcc.com.cn）

北京丰源印刷厂印刷

各地新华书店经售

*

2015年5月第一版 2015年5月北京第一次印刷

787毫米×1092毫米 16开本 13印张 292千字 2插页

印数0001-3000册 定价32.00元

前　言

TMS320F2812 具有强大的事件管理能力和嵌入式控制功能，特别适用于数据处理的测控场合，如工业自动化控制、电力电子技术应用、智能化仪器仪表及电机伺服控制系统等。为了帮助广大工程技术人员及教学人员尽快掌握 TMS320F2812 的 DSP 编程技术在控制中的应用，我们编写了本书。本书也适用于高等院校自动化专业以及电气工程与自动化、电气工程及其自动化专业本科“电机控制技术”、“新能源逆变控制技术”课程及“DSP 技术”、“电机控制技术”课程的设计，也可供电力电子与电力传动硕士研究生和从事控制系统的工程技术人员参考。

本书介绍了 DSP 芯片的结构、功能和接口原理，深入浅出地阐述了无刷直流电动机控制、永磁同步电动机控制及光伏发电逆变控制的各种基本原理和方法，以及控制所必需的常用信号检测元件，可使读者对其控制有较为系统的了解，书中系统地介绍了硬件和软件设计方法，并提供了大量的范例给读者参考，有助于读者快速地了解整个 DSP 控制系统的框架、需要设计的重点及难点。本书共分为 6 章：第 1 章为 2812 编程技术基础；第 2 章为 DSP 控制中的编程技术；第 3 章为控制系统的硬件设计；第 4 章为 DSP 控制技术应用一：无刷直流电动机控制器；第 5 章为 DSP 控制技术应用二：永磁同步电动机通用伺服控制器；第 6 章为 DSP 控制技术应用三：光伏发电逆变控制器。

本书由江西理工大学任志斌教授和苏州大学杨勇副教授编著，参加编写及程序调试的还有谢阳萍、曾彪、李家良、黄清及童稳康等。

另外，随着 CPU 芯片的快速发展，不同公司的 CPU 芯片不断推向市场，开发研究人员采用不同芯片开发控制器已成为普遍现象，但仍不妨碍在书中以 TI 公司的 DSP 芯片为例进行介绍，因为关键是内容要明白，工具可以多种多样，如作者在实验室可根据需要同时展开 TI、MICROCHIP 及 STM 的应用开发。

虽然我们在编写过程中花了不少精力，仍难免有错误与不足之处，殷切期望广大读者批评指正。

编　者

2015 年 2 月

目 录

前言

第 1 章 TMS320F2812 编程技术基础 …… 1

1.1 TMS320F28×系列芯片的结构及性能 …… 1

1.2 TMS320F2812 的存储空间及时钟 …… 5

1.2.1 存储空间 …… 5

1.2.2 CMD 文件 …… 7

1.2.3 时钟 …… 11

1.3 TMS320F2812 的中断系统 …… 14

1.3.1 外设中断介绍 …… 14

1.3.2 PIE 中断向量及其映射方式 …… 15

1.3.3 TMS320F2812 的 3 级中断机制 …… 19

1.3.4 CCS 对中断的定义、初始化及使用 …… 21

1.3.5 TMS320F2812 中断处理过程举例 …… 24

1.4 事件管理器 …… 25

1.4.1 事件管理器功能 …… 28

1.4.2 事件管理器的寄存器地址 …… 29

1.4.3 通用目的（GP）定时器 …… 31

1.4.4 全比较单元电路 …… 45

1.4.5 QEP 电路 …… 48

1.4.6 捕获单元 …… 51

1.5 DSP 的 A/D 转换器 …… 53

1.5.1 ADC 模块结构 …… 53

1.5.2 TMS320F2812 内部 ADC 的工作方式 …… 55

1.5.3 举例 …… 57

第 2 章 DSP 控制中的编程技术 …… 61

2.1 定点 DSP 的数据 Q 格式 …… 61

2.1.1 Q 格式说明 …… 61

2.1.2 电流采样值的Q格式处理 …… 62
2.2 带有死区的PWM波形 …… 64
2.3 数字PI调节器的DSP实现方法 …… 66
2.3.1 模拟PI调节器的数字化 …… 66
2.3.2 改进的数字PI算法 …… 67
2.3.3 数字PI调节器的举例 …… 68
2.4 数字测速 …… 69
2.4.1 旋转编码器 …… 69
2.4.2 数字测速方法的精度指标 …… 70
2.4.3 M法测速 …… 70
2.4.4 T法测速 …… 72
2.4.5 M/T法测速 …… 73
2.4.6 速度测量的实现 …… 75
2.4.7 例程 …… 76
2.5 电压空间矢量PWM（SVPWM）控制技术 …… 81
2.5.1 空间矢量的定义 …… 81
2.5.2 电压与磁链空间矢量的关系 …… 82
2.5.3 PWM逆变器基本输出电压矢量 …… 83
2.5.4 正六边形空间旋转磁场 …… 84
2.5.5 期望电压空间矢量的合成与实现 …… 85
2.5.6 SVPWM的三个关键问题解决 …… 86
2.5.7 SVPWM编程实现举例 …… 96
2.6 矢量变换控制技术 …… 101
2.6.1 矢量变换控制的基本思想 …… 101
2.6.2 坐标变换 …… 103

第3章 控制系统的硬件设计 …… 107
3.1 辅助电源电路 …… 108
3.1.1 VIPER22A工作原理 …… 108
3.1.2 开关电源电路设计 …… 109
3.2 功率电路设计 …… 110
3.2.1 整流电路 …… 110
3.2.2 逆变电路 …… 111

3.2.3 光耦隔离电路 …… 112
3.2.4 驱动电路 …… 113
3.3 控制电路设计 …… 114
3.3.1 DSP 外围电路 …… 114
3.3.2 电流采样电路 …… 115
3.3.3 转速检测 …… 116
3.3.4 DC 检测 …… 116
3.3.5 保护电路 …… 118
3.3.6 操作面板设计 …… 121

第 4 章 无刷直流电动机控制器 …… 125
4.1 无刷直流电动机的组成结构和工作原理 …… 125
4.1.1 无刷直流电动机的结构 …… 125
4.1.2 无刷直流电动机的霍尔传感器位置检测 …… 126
4.1.3 无刷直流电动机的工作原理 …… 130
4.2 无刷直流电动机的基本公式 …… 132
4.3 无刷直流电动机的 DSP 控制 …… 133
4.3.1 一般交流传动控制系统结构 …… 133
4.3.2 无刷直流电动机控制框图 …… 134
4.3.3 控制程序设计 …… 135
4.4 无刷直流电动机相序测定方法 …… 142
4.5 无刷直流电动机无霍尔传感器控制方法与实现 …… 143
4.5.1 采用无位置传感器控制的必要性 …… 143
4.5.2 无刷直流电动机无位置传感器控制方法 …… 143
4.5.3 无刷直流电动机无位置传感器控制原理框图 …… 144
4.5.4 控制系统软件编程设计 …… 144

第 5 章 通用伺服控制器 …… 149
5.1 控制器基本原理 …… 149
5.1.1 永磁同步电动机的数学模型 …… 150
5.1.2 永磁同步电动机矢量控制原理 …… 153
5.2 伺服控制器软件实现 …… 154
5.2.1 伺服控制器软件设计总体结构 …… 154

5.2.2 伺服控制功能模块的软件设计 …… 157
5.2.3 操作面板软件实现 …… 161

第 6 章 光伏发电逆变控制器 …… 169
6.1 光伏发电最大功率跟踪原理 …… 169
6.1.1 光伏电池的模型与特性 …… 169
6.1.2 光伏发电最大功率跟踪方法 …… 173
6.1.3 光伏发电最大功率跟踪方法的 DSP 实现 …… 176
6.2 单相光伏并网发电系统 …… 181
6.2.1 单相光伏并网发电系统的控制原理 …… 181
6.2.2 单相光伏并网发电系统的软件实现 …… 190

第1章

TMS320F2812编程技术基础

TMS320F28×系列是TI公司最新推出的数字信号处理器（DSP）芯片，是目前国际市场上最先进、功能最强大的32位定点DSP芯片。它既具有数字信号处理能力，又具有强大的事件管理能力和嵌入式控制功能，特别适用于有大批量数据处理的测控场合，如工业自动化控制、电力电子技术应用、智能化仪器仪表及电机、电机伺服控制系统等。本章将介绍TMS320F28×系列芯片的结构、性能、特点及相关编程技术。

1.1 TMS320F28×系列芯片的结构及性能

F28×系列的主要片种有TMS320F2810和TMS320F2812。两种芯片的差别是：F2812内含128K×16位的片内Flash存储器，具有外部存储器接口；而F2810仅有64K×16位的片内Flash存储器，且无外部存储器接口。其硬件特征见表1-1。

表1-1　　硬　件　特　征

特征	型　号	
	F2810	F2812
指令周期（150MHz）	6.67ns	6.67ns
SRAM（16位/字）	18K	18K
3.3V片内Flash（16位/字）	64K	128K
片内Flash/SRAM的密钥	有	有
Boot ROM	有	有
掩膜ROM	有	有
外部存储器接口	无	有
事件管理器A和B（EVA和EVB）	EVA、EVB	EVA、EVB
*通用定时器	4	4
*比较寄存器/脉宽调制	16	16
*捕获/正交解码脉冲电路	6/2	6/2

续表

特征	型　号	
	F2810	F2812
看门狗定时器	有	有
12 位的 ADC	有	有
* 通道数	16	16
32 位的 CPU 定时器	3	3
串行外围接口	有	有
串行通信接口（SCI）A 和 B	SCIA、SCIB	SCIA、SCIB
控制器局域网络	有	有
多通道缓冲串行接口	有	有
数字输入/输出引脚（共享）	有	有
外部中断源	3	3
供电电压	核心电压 1.8V I/O 电压 3.3V	核心电压 1.8V I/O 电压 3.3V
封装	128 针 PBK	179 针 GHH，176 针 PGF
温度选择①A：－40～＋85℃ S：－40～＋125℃	PBK 仅适用于 TMS	PGF 和 GHH 仅适用于 TMS
产品状况② 产品预览（PP） 高级信息（AI） 产品数据（PD）	AI （TMP）③	AI （TMP）③

① “S”是温度选择（－40～＋125℃）的特征化数据，仅对 TMS 是适用的。

② 产品预览（PP）：在开发阶段的形成和设计中与产品有关的信息，特征数据和其他规格是设计的目标。TI 保留了正确的东西，更换或者终止了一些没有注意到的产品。

高级信息（AI）：在开发阶段的取样和试制中与新产品有关的信息，特征数据和其他规格用以改变那些没有注意到的东西。

产品数据（PD）：是当前公布的数据信息，产品遵守 TI 的每项标准保修规格，但产品加工不包括对所有参数的测试。

③ TMP：最终的硅电路小片，它与器件的电气特性相一致，但是没有进行全部的品质和可靠性检测。

本书重点以 TMS320F2812 为主（以后简称为 2812），芯片的主要性能如下：

（1）高性能静态 CMOS（Static CMOS）技术。

1）150MHz（时钟周期为 6.67ns）。

2）低功耗（核心电压为 1.8V，I/O 口电压为 3.3V）。

3）Flash 编程电压为 3.3V。

（2）JTAG 边界扫描（Boundary Scan）支持。

（3）高性能的 32 位中央处理器。

1) 16位×16位和32位×32位乘且累加操作。

2) 16位×16位的两个乘且累加。

3) 哈佛总线结构（Harvard Bus Architecture）。

4) 强大的操作能力。

5) 迅速的中断响应和处理。

6) 统一的寄存器编程模式。

7) 可达4兆字的线性程序地址。

8) 可达4兆字的数据地址。

9) 代码高效（用C/C++或汇编语言）。

10) 与TMS320F24x/LF240x处理器的源代码兼容。

(4) 片内存储器。

1) 8K×16位的Flash存储器。

2) 1K×16位的OTP型只读存储器。

3) L0和L1：两块4K×16位的单口随机存储器（SARAM）。

4) H0：一块8K×16位的单口随机存储器。

5) M0和M1：两块1K×16位的单口随机存储器。

(5) 引导只读存储器（Boot ROM）4K×16位。

1) 带有软件的Boot模式。

2) 标准的数学表。

(6) 外部存储器接口（仅F2812有）。

1) 有多达1MB的存储器。

2) 可编程等待状态数。

3) 可编程读/写选通计数器（Strobe Timing）。

4) 三个独立的片选端。

(7) 时钟与系统控制。

1) 支持动态地改变锁相环的频率。

2) 带有片内振荡器。

3) 带有看门狗定时器模块。

(8) 三个外部中断引脚。

(9) 外部中断扩展（PIE）模块。可支持96个外部中断，当前仅使用了45个外部中断。

(10) 128位的密钥（Security Key/Lock）。

1) 保护Flash/OTP和L0/L1 SARAM。

2) 防止ROM中的程序被盗。

(11) 三个32位的CPU定时器。

(12) 电动机控制外围设备。

1) 两个事件管理器（EVA、EVB）。

2) 与240兼容的指令和代码。

(13) 串口外围设备。

1) 串行外围接口 (SPI)。

2) 两个串行通信接口 (SCIs)，标准的 UART 接口。

3) 改进的局域网络 (eCAN)。

4) 多通道缓冲串行接口 (McBSP) 和串行外围接口模式。

(14) 12 位的 ADC，16 通道。

1) 2×8 通道的输入多路选择器。

2) 两个采样保持器。

3) 单个的转换时间为 200ns。

4) 单路转换时间为 60ns。

(15) 最多具有 56 个独立的可编程、多用途通用输入/输出 (GPIO) 引脚。

(16) 高级的仿真特性。

1) 分析和设置断点的功能。

2) 实时的硬件调试。

(17) 开发工具。

1) ANSI C/C++编译器/汇编程序/连接器。

2) 支持 TMS320C24x/240x 的指令集。

3) 代码编辑集成环境。

4) DSP/BIOS。

5) JTAG 扫描控制器 (TI 或第三方的)。

6) 硬件评估板。

(18) 低功耗模式和节能模式。

1) 支持空闲模式、等待模式、挂起模式。

2) 停止单个外围的时钟。

(19) 封装方式。

1) 带外部存储器接口的 179 球形触点 BGA 封装。

2) 带外部存储器接口的 176 引脚低剖面四芯线扁平 LQFP 封装。

3) 没有外部存储器接口的 128 引脚贴片正方扁平 PBK 封装。

(20) 温度选择。

1) A：－40～＋85℃；

2) S：－40～＋125℃。

28x 芯片的功能框图如图 1－1 所示。

TI 公司生产的 DSP 片种还有 TMS320C2812，称为 C 系列，与 TMS320F2812 的 F 系列的差别在于：F 系列带有 Flash 存储器，而 C 系列不带 Flash 存储器；F 系列的一次性可编程 (OTP) ROM 在 C 系列中改成 ROM。若采用 C 系列编写程序，则必须在软件开发完后将代码交付给生产厂商，在出厂前将程序固化进 ROM；而 F 系列的程序存储器可以重复编程、反复擦写，在产品开发阶段使用起来比较方便。

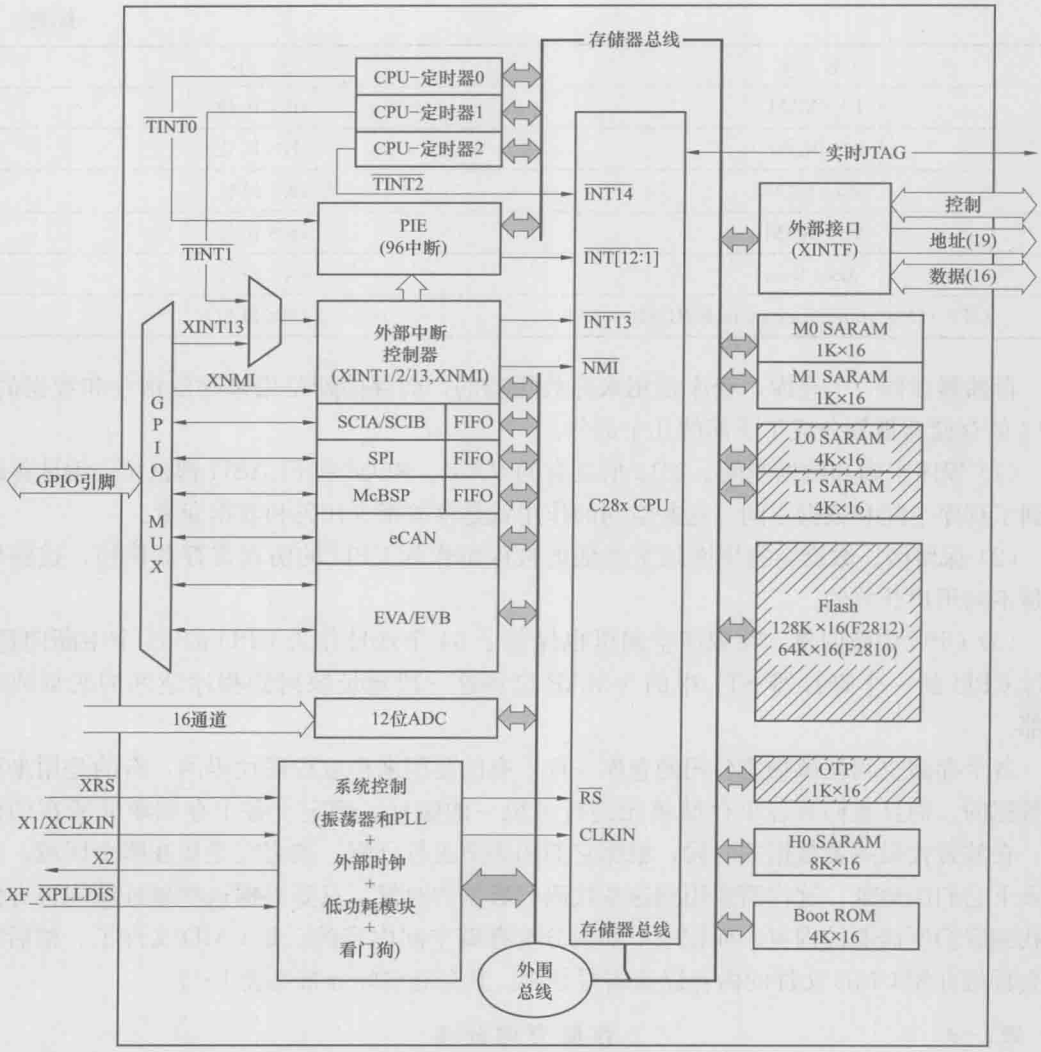

图 1-1　28x 功能框图

注　有斜线方框是代码保护的模块；＋ 器件上提供 96 个中断，45 个可用；＋ XINTF 在 F2810 上不可用。

1.2　TMS320F2812 的存储空间及时钟

1.2.1　存储空间

2812 片内集成了 RAM、ROM 和 FLASH，其具体的存储器资源见表 1-2。

表 1-2　　片内存储器资源

名　称	大　小
FLASH	128K×16 位
H0（RAM）	8K×16 位

续表

名　称	大　小
L0（RAM）	4K×16 位
L1（RAM）	4K×16 位
M0（RAM）	1K×16 位
M1（RAM）	1K×16 位
Boot Rom	4K×16 位
OTP（One Time Programmable ROM）	1K×16 位

存储器就像一个仓库，仓库是用来存放货物的，而存储器是用来存放指令和数据的。2812 的存储器被划分成了下面的几个部分。

（1）程序空间和数据空间。2812 所具有的 RAM、ROM 和 FLASH 都被统一编址并映射到了程序空间和数据空间，这些空间的作用就是存放指令代码和数据变量。

（2）保留区。数据空间里面的某些地址被保留作为 CPU 的仿真寄存器使用，这些地址是不向用户开放的。

（3）CPU 中断向量。在程序空间里也保留了 64 个地址作为 CPU 的 32 个中断向量。通过 CPU 的一个寄存器 ST1 中的 VMAP 位将这一段地址映射到程序空间的底部或者顶部。

各个存储空间就像物流公司的仓库一样，有的是用来存放程序代码的，有的是用来存放数据的。而且我们对各个存储单元进行了统一的编址，确定了各个存储单元所在的位置，在放置代码或者数据的时候，根据它们的类型进行分配，确定究竟放在哪个区域，并记录下它们的地址，这样需要用到这些代码和数据的时候，只要根据这些地址就能很方便地找到我们所需要的内容。而记录下如何分配存储空间内容的就是 CMD 文件了。稍后我们会详细介绍 CMD 文件的内容以及编写方法。其存储空间分布见表 1－3。

表 1－3　　存 储 空 间 分 布

低地址空间			
0x0000 0000	M0 矢量 RAM（VMAP=0）	0x0000 6000	外设帧 Framel（4K×16 位）
0x0000 0040	M0 SARAM（1K×16 位）	0x0000 7000	外设帧 Frame2（4K×16 位）
0x0000 0400	M1 SARAM（1K×16 位）	0x00008000	L0 SARAM（4K×16 位）
0x0000 0800	外设帧 Frame0（2K×16 位）	0x0000 9000	L1 SARAM（4K×16 位）
0x0000 0D00	PIE 向量（256×16 位，VMAP=1）	0x0000 A000	保留空间
0x0000 1000	保留空间		
高地址空间			
0x003D 7800	OTP（1K×16 位并保留 1K）	0x003F A000	保留空间
0x003D 8000	FLASH（128K×16 位）	0x003F F000	Boot Rom（4K×16，MP/$\overline{MC}$=0）
0x003F 7FF8	128 位密钥	0x003F FFC0	BROM 向量（VMAP=1，MP/$\overline{MC}$=0，ENPIE=0）
0x003F 8000	H0 SARAM（8K×16 位）		

例如，我们的程序可以放在以 00X3F800 开始的 RAM 区内，也可放在以 0X3D800 开始的 FLASH 区内，究竟放在什么地方以及其他数据的存放地址是什么，这些都是由 CMD 文件通过链接程序实现分配的。

1.2.2 CMD 文件

CMD 是 command（命令）的缩写。顾名思义，CMD 文件就是命令文件，它用来分配 ROM 和 RAM 空间，告诉链接程序怎样计算地址和分配空间。所以不同的芯片就有不同大小的 ROM 和 RAM，存放用户程序的地方也不尽相同，所以要根据芯片对文件进行修改。CMD 文件分为两部分，分别是 MEMORY 和 SECTIONS。2812 的 CMD 文件采用的是分页制，其中 PAGE0 用于存放程序空间，而 PAGE1 用于存放数据空间。我们以一个常用的 SRAM. CMD 文件为例来说明程序是如何分配地址的。

SRAM. CMD 文件如下：

```
MEMORY
{
   PAGE 0:
   PRAMH0:origin=0x3F8000,length=0x001000
   PAGE 1:/*SARAM*/
   RAMM0:origin=0x000000,length=0x000400
   RAMM1:origin=0x000400,length=0x000400
   /*Peripheral Frame 0:*/
   DEV_EMU:origin=0x000880,length=0x000180
   FLASH_REGS:origin=0x000A80,length=0x000060
   CSM:origin=0x000AE0,length=0x000010
   XINTF:origin=0x000B20,length=0x000020
   CPU_TIMER0:origin=0x000C00,length=0x000008
   CPU_TIMER1:origin=0x000C08,length=0x000008
   CPU_TIMER2:origin=0x000C10,length=0x000008
   PIE_CTRL:origin=0x000CE0,length=0x000020
   PIE_VECT:origin=0x000D00,length=0x000100
   /*Peripheral Frame 1:*/
   ECAN_A:origin=0x006000,length=0x000100
   ECAN_AMBOX:origin=0x006100,length=0x000100
   /*Peripheral Frame 2:*/
   SYSTEM:origin=0x007010,length=0x000020
   SPI_A:origin=0x007040,length=0x000010
   SCI_A:origin=0x007050,length=0x000010
   XINTRUPT:origin=0x007070,length=0x000010
   GPIOMUX:origin=0x0070C0,length=0x000020
   GPIODAT:origin=0x0070E0,length=0x000020
```

```
 ADC:origin=0x007100,length=0x000020
 EV_A:origin=0x007400,length=0x000040
 EV_B:origin=0x007500,length=0x000040
 SPI_B:origin=0x007740,length=0x000010
 SCI_B:origin=0x007750,length=0x000010
 MCBSP_A:origin=0x007800,length=0x000040
 /*CSM Password Locations*/
 CSM_PWL:origin=0x3F7FF8,length=0x000008
 /*SARAM*/
 DRAMH0:origin=0x3F9000,length=0x001000
}
SECTIONS
{
 /*Allocate program areas:*/
 .reset:>PRAMH0,PAGE=0
 .text:>PRAMH0,PAGE=0
 .cinit:>PRAMH0,PAGE=0
 /*Allocate data areas:*/
 .stack:>RAMM1,PAGE=1
 .bss:>DRAMH0,PAGE=1
 .ebss:>DRAMH0,PAGE=1
 .const:>DRAMH0,PAGE=1
 .econst:>DRAMH0,PAGE=1
 .sysmem:>DRAMH0,PAGE=1
 /*Allocate Peripheral Frame 0 Register Structures:*/
 DevEmuRegsFile:>DEV_EMU,PAGE =1
 FlashRegsFile:>FLASH_REGS,PAGE=1
 CsmRegsFile:>CSM,PAGE=1
 XintfRegsFile:>XINTF,PAGE=1
 CpuTimer0RegsFile:>CPU_TIMER0,PAGE=1
 CpuTimer1RegsFile:>CPU_TIMER1,PAGE=1
 CpuTimer2RegsFile:>CPU_TIMER2,PAGE=1
 PieCtrlRegsFile:>PIE_CTRL,PAGE=1
 PieVectTable:>PIE_VECT,PAGE=1
 /*Allocate Peripheral Frame 2 Register Structures:*/
 ECanaRegsFile:>ECAN_A,PAGE=1
 ECanaMboxesFile:>ECAN_AMBOX PAGE=1
 /*Allocate Peripheral Frame 1 Register Structures:*/
 SysCtrlRegsFile:>SYSTEM,PAGE=1
 SpiaRegsFile:>SPI_A,PAGE=1
 SciaRegsFile:>SCI_A,PAGE=1
```

```
XIntruptRegsFile:>XINTRUPT,PAGE=1
GpioMuxRegsFile:>GPIOMUX,PAGE=1
GpioDataRegsFile:>GPIODAT PAGE=1
AdcRegsFile:>ADC,PAGE=1
EvaRegsFile:>EV_A,PAGE=1
EvbRegsFile:>EV_B,PAGE=1
ScibRegsFile:>SCI_B,PAGE=1
McbspaRegsFile:>MCBSP_A,PAGE=1
/*CSM Password Locations*/
CsmPwlFile:>CSM_PWL,PAGE=1
}
```

上述文件中，MEMORY 和 SECTIONS 是命令文件中最常用的两条伪指令。MEMORY 伪指令用来表示实际存在目标系统中的可以使用的存储器范围，在这里每个存储器都有自己的名字、起始地址和长度。SECTIONS 伪指令是用来描述输入端是如何组合到输出端内的。下面我们来具体分析。

在 MEMORY 伪指令中：

```
PAGE 0 :
PRAMH0:origin=0x3F8000,length=0x001000
PAGE 1:/*SARAM* /
RAMM0:origin=0x000000,length=0x000400
RAMM1:origin=0x000400,length=0x000400
  /*Peripheral Frame 2:*/
  ......
EV_A:origin=0x007400,length=0x000040
......
DRAMH0:origin=0x3F9000,length=0x001000
```

定义存储器 PRAH0 在 PAGE0，起始地址和长度分别为 0x3F8000 和 0x001000。定义 RAMM0、RAMM1 及 EV _ A 等在 PAGE1，以及各自的起始地址和长度。

在 SECTION 伪指令中：

```
/*Allocate program areas:*/
.reset:>PRAMH0,PAGE=0
.text:>PRAMH0,PAGE=0
.cinit:>PRAMH0,PAGE=0
```

其中：.text 为所有可以执行的代码和常量；.cinit 为全局变量和静态变量的 C 初始化记录；.reset 为复位中断向量，这些都放在 origin＝0x3F8000 和 length＝0x001000 中。