具有排错功能的 TMS320F2812DSP 与 PC 机的串口通信设计

杨 茜 李文伟 何家远 朱艳芳 付 士 白 雷
(湖北三江航天红峰控制有限公司，孝感 432000)

摘要：根据串口通信规则设计了 TMS320F2812DSP 与 PC 机之间的串口通信协议，采用 SCI 模块的 FIFO 堆栈模式进行接收数据操作，实现数据的实时更新及数据接收过程中的排错功能，保证了数据的实时性和准确性。

关键词：串口通信；通信协议；排错功能

Serial Communication Design with Function of Trouble Shooting on TMS320F2812DSP and PC

Yang Xi Li Wenwei He Jiayuan Zhu Yanfang Fu Shi Bai Lei
（Hubei Sanjiang Space Honfeng Control Co. Ltd., Xiaogan 432000）

Abstract：The SCI communication protocol on TMS320F2812DSP and PC was designed according to the protocol of serial port communication. To receive data by using FIFO stack mode of SCI module, the function of data real time updating and trouble shooting was realized, and the instantaneity and veracity of data were ensured.

Key words：serial communication；communication protocol；trouble shooting

1 引言

DSP全称Digital Signal Processor，为美国德州仪器公司(TI)开发研制的数字信号处理器的统称。TMS320F2812是TI公司推出的C2000平台上的定点32位DSP芯片，适用于电机控制、工业控制等。

本文根据串口通信规则设计了DSP与PC机之间的串口通信协议，采用SCI模块的增强型FIFO堆栈模式进行接收数据操作，同时实现了数据接收过程中的排错功能，保证了数据传输的实时性与准确性。PC机采用串口调试工具进行数据接收与发送，DSP控制软件采用Ram单板调试模式，并通过CCS中的变量观察窗对接收数组变量进行实时监控。

2 SCI 通信原理

要保证通信正常进行，上位机与DSP须设定为相同的波特率。DSP的两个SCI模块均具有两个8位的波特率寄存器，SCIHBAUD与SCILBAUD，可以通过编程实现64K种不同波特率的设定。具体计算公式如下：

$$BRR = \frac{LSPCLK}{SaAsynchronousBaud \times 8} - 1 \qquad (1)$$

其中，SaAsynchronousBau为SCI传输波特率，本文设定为614400bps，LSPCLK为低速外设时钟，本文外部晶振为TMS320F2812DSP提供30MHz的时基，锁相环控制寄存器PLLCR的值设定为8，可以计算出送至CPU的时钟为120MHz，低速外设时钟预定标寄存器LOSPCP取值为2，即低速外设时钟为CPU时钟4分频30MHz。依式（1）计算出BRR的值为0x05。

3 SCI 通信协议

作者简介：杨茜（1987-），硕士，检测技术与自动化装置专业；研究方向：车载网络技术、总线通信技术。
收稿日期：2014-01-09

F2812的SCI模块使用的是标准非归零（NRZ）数据格式，能通过寄存器SCICCR设置SCI的数据格式，本设计通过RS-422串口接收来自PC机串口发送的26个字节数据，接收数据格式如图1所示。

0x55 位0	0xAA 位1	位2	位3	位4	位5	位6	位7	位8	位9	位10	位11	位12
位13	位14	位15	位16	位17	位18	位19	位20	位21	位22	位23	位24	校验和 位25

图 1　接收数据格式

本文定义传输数据为unsigned char类型。主机发送给DSP的数据包含了一个报文头0xAA55、23个unsigned char数据位、一个校验和位，其中校验和为数据包中所有数据之和。传输开始后，DSP判断报文头正确之后进行数据接收，按照按位相加方式计算校验和并判断是否与接收数据包中的校验和相等，否则丢弃该数据包。

4　排错功能

F2812的SCI模块FIFO堆栈深度为16级，本文中需要接收的数据长度为26，大于堆栈深度。为保证数据的实时性及准确性，设定了两个数组，一个数据buffer_1[52]用于实时地记录所接收的数据，buffer_2[26]用于存放取自数组buffer_1[52]的一组正确接收的数据，并能实时更新。DSP接收数据采用FIFO中断实现，为保证数据正常接收，设定SCI模块的SCIFFRX寄存器的位RXFFIL为13，即当上位机向DSP发送数据达13位的时候，产生一次SCI接收中断，取出堆栈中的13位数，读取并置入buffer_1[52]中，剩下的13个数在下次中断时读出并置入buffer_1[52]中。循环查找buffer_1[52]中是否有符合要求的26个数据，并将置入buffer_2[26]。

需要申请两个数据，一个读指针PR，一个写指针PW。指针接收数据之后的变换规则如图2所示。

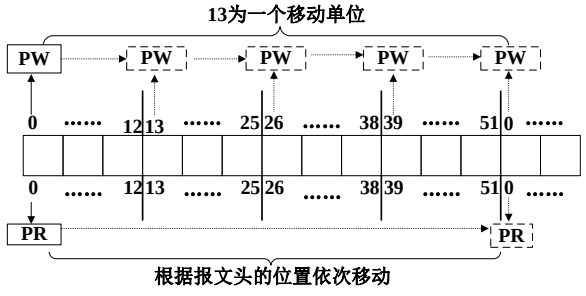


图 2　接收数据移动规则

图中写指针在SCI中断发生时接收13个数据，其位置往后移动13位，其在循环数据接收过程中，每次收到数据为26个字节，因此PW的位置总在0、26上变化。PR在程序循环查找过程中，查找到报文头之后，会自动移动到0x55处，其位置随机分布。

具有排错功能的数据接收流程图如图3所示。

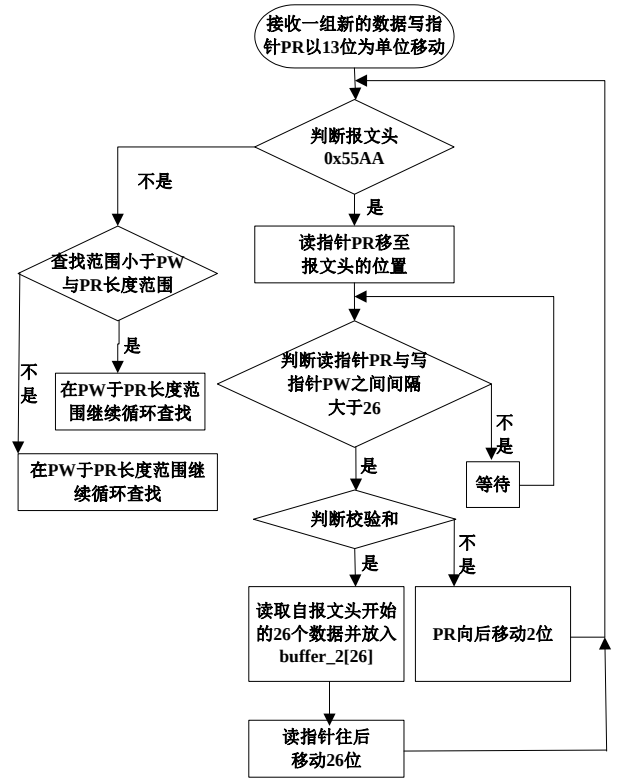


图 3　接收数据流程

5　程序设计

程序的整体思路如下所示：

a. 初始化系统，为系统分配时钟，处理看门狗电路等。

SysCtrlRegs.PLLCR=0x8; //系统时钟为30*8/2=120MHz

SysCtrlRegs.LOSPCP.all=0x0002;//LSPCLK= SYSCLKOUT / 4=120/4=30MHz

SysCtrlRegs.PCLKCR.bit.SCIAENCLK=1; //使能SCI时钟

b. 初始化GPIO，将SCI得引脚SCITXDA和SCIRXDA设定为功能引脚。

GpioMuxRegs.GPFMUX.all=0x0030;//设置SCI的发送接收引脚

c. 初始化SCI模块，设定通信数据格式，使能SCI

FIFO模式，并配置相关的寄存器。

SciaRegs.SCIFFTX.all=0xE040;//使能FIFO操作

SciaRegs.SCIFFRX.all=0x606d;//接收13个数时产生SCI中断

SciaRegs.SCICCR.all =0x0007;//8个数据位

SciaRegs.SCIHBAUD =0x0000;

SciaRegs.SCILBAUD =0x0005;// 波特率 614400 b/s

SciaRegs.SCICTL1.all =0x0023;//

d. SCI等待接收数据，一旦数据到达，SCI将其写入接收FIFO，接收完毕，将数据按照一定规则发送回上位机。

interrupt void SCIRXINTA_ISR(void) //接收中断函数

{int i;

for(i=PW;i<PW+13;i++)//从写指针当前位置开始依次写入新的数据到buffer_1[52]中

{buffer_1 [i] = SciaRegs.SCIRXBUF.all ;}

PW=(PW+13)%52;//指针移至下一个数据存放起始位置

asm ("ESTOP0");}

控制循环函数实现26个数据的正确读取。接收发送数据的程序设计如下所示：

interrupt void cpu_timer0_isr(void)

{int j, p, q, m;

for (i=PR;i<PR+(PW+52-PR)%52-1;(i++)%52)

{if((buffer_1[i]==0x55)&&(buffer_1[i+1]==0xAA))

{PR=i;//找到了报文头

if ((PW+52-PR)%52 >=26)//判断头与尾的长度

{for (j=2;j<25,j++)

{checksum = checksum + buffer_1[(PR+j)%52];//计算校验和}

if (checksum == buffer_1[(PR+j)%52])//校验和满足要求

{//将26个数据全部放入buffer_2[26]中

for (p=0;p<26;p++)

{buffer_2[p] = buffer_1[(p+PR)%52];}

i=(p+PR)%52; break;}

else {PR = (PR+2)%52;//把数据头跳过去}}

else {//等待下一包数据}}}

PR = i;}

6 硬件连接

采用PCI接口的8路串口卡BST23108设置异步RS422模式，设计中采用了其中的4路信号，分别为管脚1：IN0-（第0路RS422接收负）、管脚2：IN0+（第0路RS422j接收正）、管脚3：OUT0-（第0路RS422发送负）、管脚4：OUT0+（第0路RS422j发送正）。将装有DSP2812的控制板与串口卡相连接，按照设定的通信协议进行数据通信。

7 排错功能验证

设计三组数据验证所设计排错功能的正确性。具体数据分配如表1～表3所示。

表 1 验证数组 1，2

第一组		第二组	
位	数据（Hex）	位	数据（Hex）
0	0x55	0	0x55
1	0xAA	1	0xAA
2～24	均为 0x02	2～24	均为 0x01
25	0x2E	25	0x17

表 2 验证数组 3

第三组			
位	数据（Hex）	位	数据（Hex）
0	0x00	31	0x03
1～2	0x00	32	0x04
3	0x55	33	0x05
4	0xAA	34	0x06
5～27	0x01	35	0x07
28	0x17	36	0x08
29	0x01	37	0x09
30	0x02	38	0x10

当验证数组由串口工具发送给DSP下位机时，可以通过CCS监控功能实时查看数组变量值。图4为验证数组1发送至DSP时所采集到的数据信息，buffer_2读取了buffer_1数组中第0位至第25位的值。

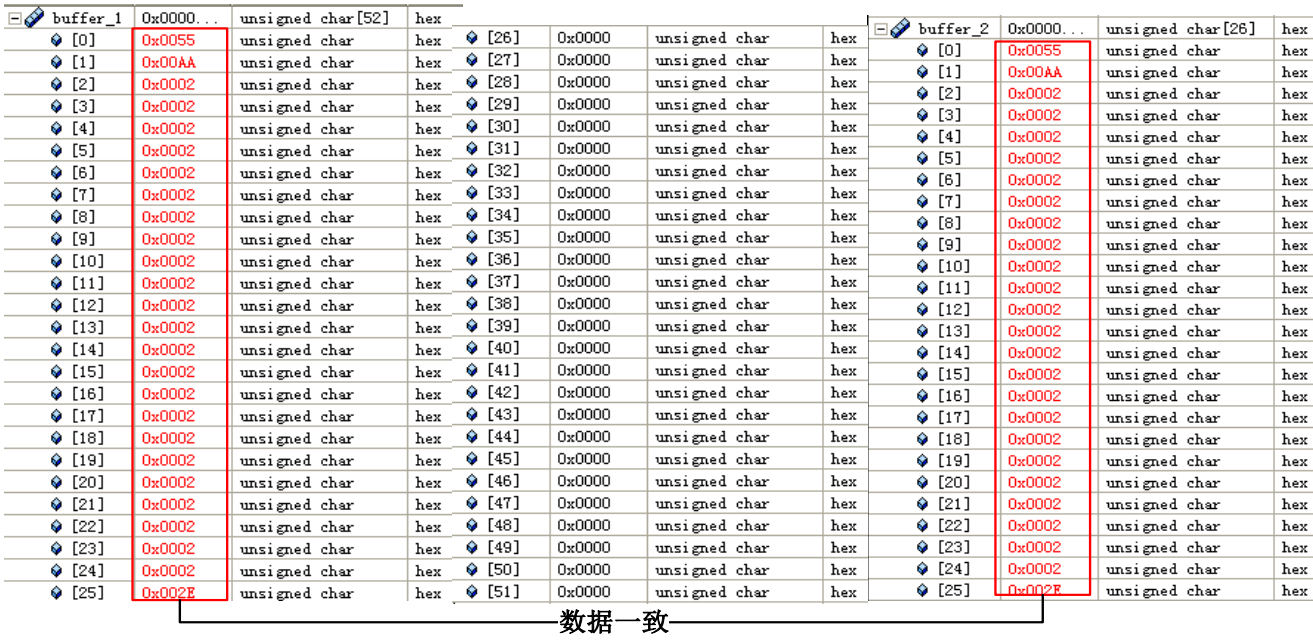


图 4　第一组接收数据对照

当第二次发送另一组报文头、校验和均满足要求的数组时，采集到的数据如图5所示。图中buffer_1数组第26位至第51位显示了本次更新数据，该组数据在buffer_2中同步更新。buffer_1第0位至第25位仍然保留上次数据值。

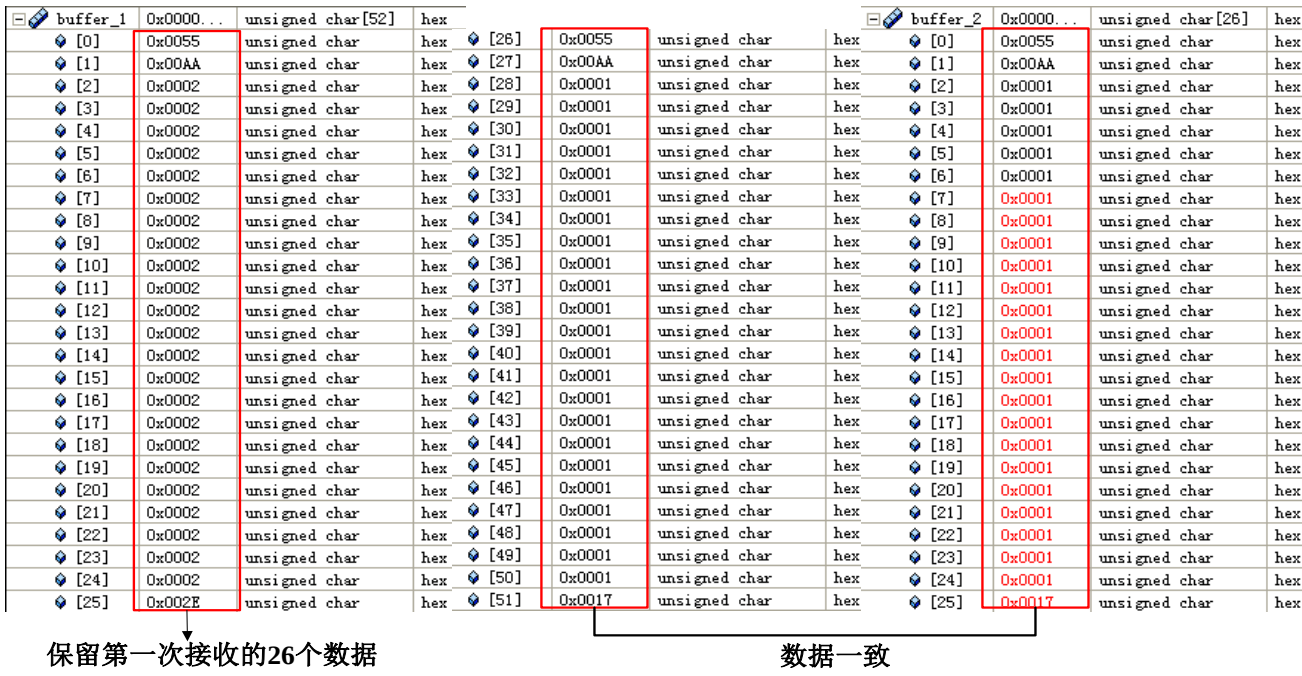
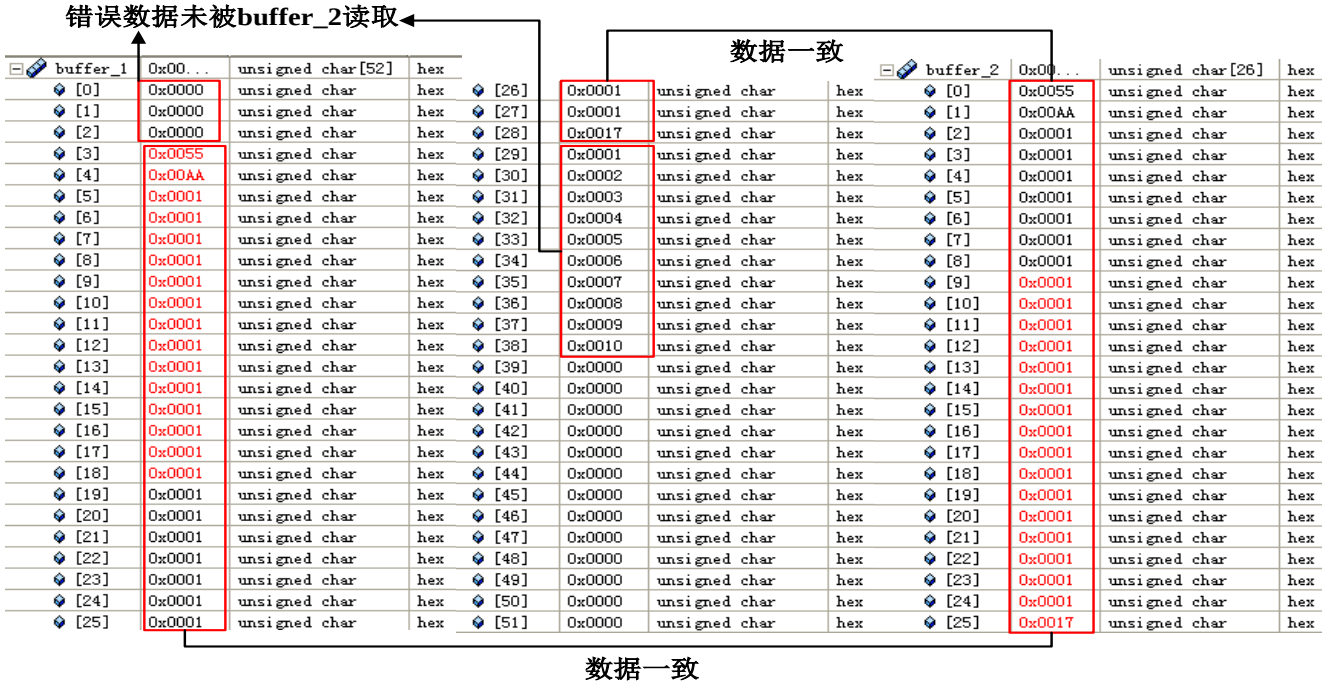


图 5　第二组接收数据对照

第三组数据发送错误数据如表2所示，数据实时采集情况如图6所示。由串口工具发送的39个数据依次占据了buffer_1的第0位至第38位，其中第3位至第28位为一组报文头、校验和均符合要求的数据。第0位至第2位和第29位至38位为不符合要求的数据。buffer_2实时显示了取自buffer_1的自第3位到第28位

的数据，自动丢弃了所有不符合要求的数据。



buffer_1	0x00...	unsigned char[52]	hex					buffer_2	0x00...	unsigned char[26]	hex
[0]	0x0000	unsigned char	hex	[26]	0x0001	unsigned char	hex	[0]	0x0055	unsigned char	hex
[1]	0x0000	unsigned char	hex	[27]	0x0001	unsigned char	hex	[1]	0x00AA	unsigned char	hex
[2]	0x0000	unsigned char	hex	[28]	0x0017	unsigned char	hex	[2]	0x0001	unsigned char	hex
[3]	0x0055	unsigned char	hex	[29]	0x0001	unsigned char	hex	[3]	0x0001	unsigned char	hex
[4]	0x00AA	unsigned char	hex	[30]	0x0002	unsigned char	hex	[4]	0x0001	unsigned char	hex
[5]	0x0001	unsigned char	hex	[31]	0x0003	unsigned char	hex	[5]	0x0001	unsigned char	hex
[6]	0x0001	unsigned char	hex	[32]	0x0004	unsigned char	hex	[6]	0x0001	unsigned char	hex
[7]	0x0001	unsigned char	hex	[33]	0x0005	unsigned char	hex	[7]	0x0001	unsigned char	hex
[8]	0x0001	unsigned char	hex	[34]	0x0006	unsigned char	hex	[8]	0x0001	unsigned char	hex
[9]	0x0001	unsigned char	hex	[35]	0x0007	unsigned char	hex	[9]	0x0001	unsigned char	hex
[10]	0x0001	unsigned char	hex	[36]	0x0008	unsigned char	hex	[10]	0x0001	unsigned char	hex
[11]	0x0001	unsigned char	hex	[37]	0x0009	unsigned char	hex	[11]	0x0001	unsigned char	hex
[12]	0x0001	unsigned char	hex	[38]	0x0010	unsigned char	hex	[12]	0x0001	unsigned char	hex
[13]	0x0001	unsigned char	hex	[39]	0x0000	unsigned char	hex	[13]	0x0001	unsigned char	hex
[14]	0x0001	unsigned char	hex	[40]	0x0000	unsigned char	hex	[14]	0x0001	unsigned char	hex
[15]	0x0001	unsigned char	hex	[41]	0x0000	unsigned char	hex	[15]	0x0001	unsigned char	hex
[16]	0x0001	unsigned char	hex	[42]	0x0000	unsigned char	hex	[16]	0x0001	unsigned char	hex
[17]	0x0001	unsigned char	hex	[43]	0x0000	unsigned char	hex	[17]	0x0001	unsigned char	hex
[18]	0x0001	unsigned char	hex	[44]	0x0000	unsigned char	hex	[18]	0x0001	unsigned char	hex
[19]	0x0001	unsigned char	hex	[45]	0x0000	unsigned char	hex	[19]	0x0001	unsigned char	hex
[20]	0x0001	unsigned char	hex	[46]	0x0000	unsigned char	hex	[20]	0x0001	unsigned char	hex
[21]	0x0001	unsigned char	hex	[47]	0x0000	unsigned char	hex	[21]	0x0001	unsigned char	hex
[22]	0x0001	unsigned char	hex	[48]	0x0000	unsigned char	hex	[22]	0x0001	unsigned char	hex
[23]	0x0001	unsigned char	hex	[49]	0x0000	unsigned char	hex	[23]	0x0001	unsigned char	hex
[24]	0x0001	unsigned char	hex	[50]	0x0000	unsigned char	hex	[24]	0x0001	unsigned char	hex
[25]	0x0001	unsigned char	hex	[51]	0x0000	unsigned char	hex	[25]	0x0017	unsigned char	hex

图 6　错误数据接收数据对照

8　结束语

针对串口通信过程中数据包错误传输的问题，根据需求设计串口通信协议架构，完成串口通信排错功能流程的设计。利用 SCI 的 FIFO 堆栈模式，实现了数据的正确传输。通过串口调试工具发送设定好的数据，并在 CCS 数据监控栏实时地显示并分析测试结果。证明了所设计串口数据传输排错功能的正确性。该方案可以保证数据传输的实时性和准确性，具有良好的工程应用前景。

参考文献

1　顾卫钢. 手把手教你学 DSP——基于 TMS320X281x. 北京：北京航空航天大学出版社，2011

2　TMS320F281x Datasheet. Texas Instruments, 2003

3　TMS320F281x Assembly Language Tools User's Guide. Texas instruments, 2002

4　TMS320F28x Boot ROM Reference Guide (Rev. A). Texas Instruments, 2003

5　TMS320F28xx 和 TMS320F28xxx DSCs 的硬件设计指南. Texas Instruments，2008

6　TMS320F28xx28xxx DSCs 模拟接口设计综述.Texas Instruments，2008

7　Running an Application from Internal Flash Memory on the TMS320F28xx DSP. Texas Instruments, 2008

8　苏奎峰，吕强，等. TMS320F2812 原理与开发. 电子工业出版社，2005

9　张卫宁. TMS320C28x 系列 DSP 的 CPU 与外设（下）. 北京：清华大学出版社，2005

10　孙丽明. TMS320F2812 原理及其 C 语言程序开发. 北京：清华大学出版社，2008