doi: 10.7690/bgzdh.2020.05.006

基于 DSP TMS320F28335 的外扩 Flash 存储器

曲　鹏，周应旺，许　奔
（南京模拟技术研究所发动机事业部，南京 210006）

摘要：为存储无人机飞行过程中发动机状态数据，设计一款基于 DSP TMS320F28335 的外扩 Flash 存储器。以 TMS320F28335 和 SST39VF1601 芯片为对象，根据 DSP 外部接口和 Flash 存储器的特点，运用 XZCS6 区进行外部 Flash 扩展及硬件设计，对 DSP 外扩与 Flash 存储器进行擦除、读、写等操作软件设计，并通过测试实例验证。验证结果表明，该设计有一定的实用价值。

关键词：DSP；TMS320F28335；Flash 存储器
中图分类号：TP333　　**文献标志码：**A

External Flash Memory Based on DSP TMS320F28335

Qu Peng, Zhou Yingwang, Xu Ben
(*Engine Department*, *Nanjing Research Institute of Simulation Technology*, *Nanjing* 210006, *China*)

Abstract: In order to store the data of the engine state during the flight of the UAV, an external Flash memory based on DSP TMS320F28335 was designed. Taking TMS320F28335 and SST39VF1601 chips as the object, according to the characteristics of DSP external interface and Flash memory, XZCS6 area is used for the external Flash expansion and hardware design, and the operation software of DSP external expansion and Flash memory is designed for erasing, reading, writing, and so on, and the test case is verified. The verification results show that the design has some practical value.

Keywords: DSP; TMS320F28335; Flash memory

0　引言

发动机状态数据能够准确地反映发动机工作状态，对分析发动机性能具有重大意义。为了存储无人机飞行全过程发动机状态数据，保证不丢失掉电后数据，提出扩展外部 Flash 存储器。Flash 存储器是一种高密度、非易失性电可擦写的存储器，具有成本低、体积小、质量轻、功耗低及可多次擦写等优点，适合在尺寸较小的控制器中使用。笔者以 TI 公司的 DSP TMS320F28335 和 SST 公司的 SST39VF1601 NOR Flash 存储器为基础，详细介绍外扩 Flash 存储器的硬件设计、软件开发，实现对 Flash 存储器进行擦除、读、写操作。

1　硬件接口电路设计

笔者采用 TI 公司的一款 32 位浮点型数字信号处理器 TMS320F28335（简称 F28335）和 SST 公司的 Flash 存储器 SST39VF1601。F28335 采用哈佛流水线总线结构[1-2]，高性能静态 CMOS 技术，主频可达 150 MHz，内核采用 1.9 V 供电，可用 C 语言和汇编语言进行高效编程，具有精度高、成本低、功耗小、外设集成度高、数据及程序存储容量大和 A/D 转换精度高速度快等优点，被广泛应用于军工行业中。

F28335 的外部存储区域被分为 XZCS0（区域 0）、XZCS6（区域 6）和 XZCS7（区域 7）。XZCS0、XZCS6 和 XZCS7 对应的地址范围分别为 0x004000-0x004FFF 、 0x100000-0x1FFFFF 和 0x200000-0x2FFFFF，各区域可存储空间大小分别为 4Kx16、1Mx16 和 1Mx16。F28335 外部存储器接口（XINTF）包括 20 位地址线、16 位（最大 32 位）数据线、3 个片选控制线及读写控制线，其中 3 个片选线映射到 3 个存储区域，用户可根据需求选择某一区域进行扩展[1]。

SST39VF1601 存储空间大小为 1Mx16：按扇区可以划分为 512 个扇区，每个扇区大小为 2Kx16；按块可以划分为 32 个块区，每个块区大小为 32 Kx16，供电电压 2.7～3.6 V，低功耗，工作电流 9 mA，静态电流 3 μA，可擦写 10 万次，数据保存时间超过 100 a，访问时间 70 ns，快速扇区、块区擦除时间 18 ms，快速片区擦除时间 40 ms，快速单字编程时间 7 μs，可对扇区或块区进行单独操作而不相互影响。

收稿日期：2019-12-20；**修回日期：**2020-01-07
作者简介：曲　鹏（1982—），男，山东人，硕士，工程师，从事微型涡喷发动机控制系统研究。E-mail: blueghostqu@163.com。

SST39VF1601 芯片引脚介绍：CE 为片选引脚；RESET 为复位引脚，低电平有效，复位后为读操作模式；WE 为写允许控制线，低电平有效；OE 为读允许控制线，低电平有效；WP 为写保护，低电平禁止写入和擦除，允许读取；A0～A19 为地址线；DQ0～DQ15 为数据线。

F28335 具有 20 条地址线，可寻址 1M 字异步存储器空间。笔者选用存储容量为 1M 字的外部存储器，采用 XZCS6 区进行外部 Flash 扩展，不仅能满足使用要求，而且可以对存储器全地址进行访问，无需进行分页处理。TMS320F28335 与 SST39VF1601 电路连接原理如图 1 所示。

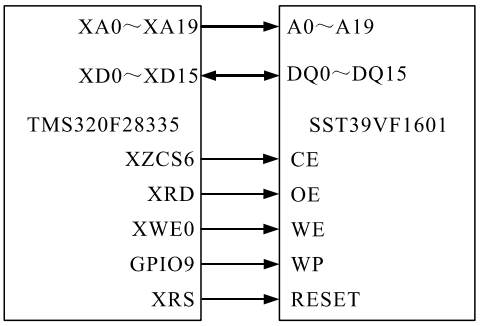

图 1 TMS320F28335 与 SST39VF1601 电路连接原理

如图所示，将 DSP 的 20 条地址线和 16 条数据线分别与 Flash 的地址线、数据线相连接，片选信号线 XZCS6、读、写控制线分别与 Flash 的 CE、OE 和 WE 相连接。GPIO9 作为写保护信号输出的引脚与 Flash 的 WP 相连，当向 Flash 写入或擦除数据时，该脚输出高电平，读取数据时则该脚输出低电平，默认状态为低电平。DSP 复位引脚 XRS 与 Flash 复位引脚 RESET 相连，目的是当产生复位信号时，Flash 设置为读取状态，保持已储存的数据不被删除。

2 软件设计

Flash 可工作在读模式、写模式和擦除模式：当 CE、OE、WP 为低电平，WE 为高电平时，工作在读模式，可直接将 Flash 地址中的数据取出；当 Flash 工作在写模式时，需向 2 个特定的地址写入 3 个特定的数值[3-4]（如表 1 所示），随后将要写入的数据直接写到某个地址内，但在写入之前要先进行擦除，否则无法正常写入数据。根据 2.2 节可知：SST39VF1601 有 3 种擦除方式，不仅擦除时间不同，而且擦除时向特定地址写入的数值也不同。

表 1 Flash 操作指令

模式	总线周期	第 1	第 2	第 3	第 4	第 5	第 6
写模式	地址	5555H	2AAAH	5555H	WA		
	数据	AAH	55H	A0H	Data		
扇区擦除模式	地址	5555H	2AAAH	5555H	5555H	2AAAH	SA
	数据	AAH	55H	80H	AAH	55H	30H
块擦除模式	地址	5555H	2AAAH	5555H	5555H	2AAAH	BA
	数据	AAH	55H	80H	AAH	55H	50H
片擦除模式	地址	5555H	2AAAH	5555H	5555H	2AAAH	5555H
	数据	AAH	55H	80H	AAH	55H	10H

注：WA 表示被写入的 Flash 地址；Data 表示被写入的数据；SA 表示扇区地址，BA 表示块区地址。

该存储器采用片擦除，周期读、写操作方式，当存储器空间写满或读取完毕后将停止读、写操作，软件开发环境为 CCStudiov3.3，使用的开发语言为 C 语言。

2.1 XINTF 初始化

XINTF 初始化主要是对外部接口时钟（XTIMCKL）、数据读\写脉冲持续时间、采样模式、数据总线位数以及复用 GPIO 口作为地址线、数据线、控制信号线等的配置[5]。

XINTF 初始化函数如下：

```
void InitXintf(void)    //XINTF 初始化函数
{
 EALLOW;
 XintfRegs.XINTCNF2.bit.XTIMCLK = 1;    //外部接口时钟=系统时钟/2
 XintfRegs.XINTCNF2.bit.WRBUFF = 0; // 无写缓冲
 XintfRegs.XTIMING6.bit.XWRLEAD = 3; // 写访问周期建立等待状态为 3 个 XTIMCKL 周期
 XintfRegs.XTIMING6.bit.XWRACTIVE = 7;// 写访问周期激活等待状态为 7 个 MTIMCKL 周期
 XintfRegs.XTIMING6.bit.XWRTRAIL = 3; // 写访问周期跟踪等待状态为 3 个 XTIMCKL 周期
 XintfRegs.XTIMING6.bit.XRDLEAD = 3; // 读访问周期建立等待状态为 3 个 XTIMCKL 周期
 XintfRegs.XTIMING6.bit.XRDACTIVE = 7;// 读访问周期激活等待状态为 7 个 MTIMCKL 周期
 XintfRegs.XTIMING6.bit.XRDTRAIL = 3; // 读访问周期跟踪等待状态为 3 个 XTIMCKL 周期
```

```
XintfRegs.XTIMING6.bit.X2TIMING=1;  // 读 / 写建立、激活、跟踪等待状态周期扩大 2 倍
XintfRegs.XTIMING6.bit.USEREADY=1;  // 采样 XREADY 信号
XintfRegs.XTIMING6.bit.READYMODE=1;  // 异步采样
XintfRegs.XTIMING6.bit.XSIZE=3; //16 位数据总线模式
GpioCtrlRegs.GPCMUX1.bit.GPIOn=3; // n=64～79 配置为数据线 XD15～XD0
GpioCtrlRegs.GPBMUX1.bit.GPIOn=3; // n=40～47 配置为地址线 XA0～XA7
GpioCtrlRegs.GPCMUX2.bit.GPIOn = 3;
//n=80～87 配置为地址线 XA8～XA15
GpioCtrlRegs.GPBMUX1.bit.GPIO39=3;//XA16
GpioCtrlRegs.GPAMUX2.bit.GPIO31=3;//XA17
GpioCtrlRegs.GPAMUX2.bit.GPIO30=3;//XA18
GpioCtrlRegs.GPAMUX2.bit.GPIO29=3;//XA19
GpioCtrlRegs.GPBMUX1.bit.GPIO38=3;// 写控制线
GpioCtrlRegs.GPAMUX2.bit.GPIO28=3;// 片选控制线
EDIS;
}
```

2.2　片擦除程序设计

笔者只对片擦除操作进行介绍，扇区与块区擦除操作与此类似。如表 1 所示，在第 1 个总线周期向 0x5555H 地址写入 0xAAH，第 2 个总线周期向 2AAAH 地址写入 55H，依次类推，按顺序完成 6 个总线周期操作后，Flash 存储器 1Mx16 空间的数据被全部擦除，其状态为 0xFFFF，擦除的起始地址为 0x100000，尾地址为 0x1FFFFF。

编写函数体前应先定义如下 2 个全局变量，该变量在片擦除函数、写操作函数与读操作函数中均适用。

```
#define WP GpioDataRegs.GPADAT.bit.GPIO9// 写保护 低电平禁止写入和擦除
unsigned int *FlashStartAdd = (unsigned int *)0x100000; //外部 Flash 起始地址
```

Flash 片擦除程序代码如下所示：

```
void ChipErase(void)
{
WP = 1; //擦除操作时为高电平
*(FlashStartAdd + 0x5555) = 0xAA;
*(FlashStartAdd + 0x2AAA) = 0x55;
*(FlashStartAdd+ 0x5555) = 0x80;
*(FlashStartAdd+ 0x5555) = 0xAA;
*(FlashStartAdd + 0x2AAA) = 0x55;
*(FlashStartAdd+ 0x5555) = 0x10;
}
```

为验证擦除操作的正确性，笔者对擦除操作前后存储空间数据进行对比，在 CCS 中点击 vies—>memory 查看寄存器，在地址栏中如输入 0x100000，可看到片擦除前、后 Flash 存储器中数据分别如图 2、图 3 所示。

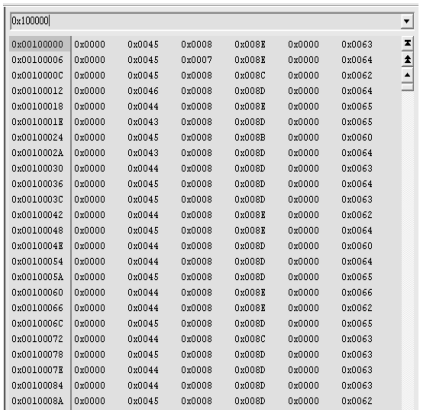

图 2　片擦除前 Flash 中数据

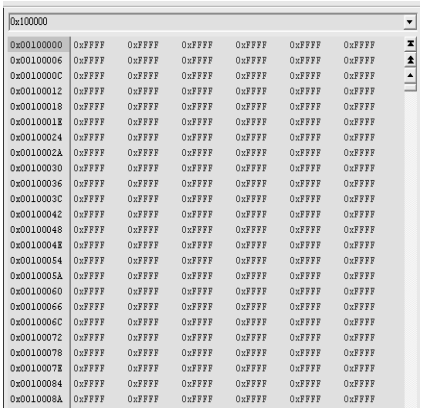

图 3　片擦除后 Flash 中数据

2.3　写操作程序设计

如表 1 所示，外部 Flash 写操作时，需在前 3 个总线周期向特定地址写入特定的数值，第 4 个总线周期将要保存的数据直接写入某个地址。笔者设计的测试实例如下：每 10 ms 保存一次数据，每次写入 6 个字，为便于观察结果，设定每个周期写入的数据为 0x1111、0x2222、0x3333、0x4444、0x5555、0x6666。

Flash 写操作程序代码如下：

```
Uint16 FlashWrite(Uint32 DataStart, Uint32 AddStart, Uint16 Length) //源地址，目的地址，地址长度
```

```
{
  Uint32 i=0;
  WP = 1; //写操作时为高电平
  for(i = 0; i < Length; i++)
  {
   *(FlashStartAdd + 0x5555) = 0xAA;
   *(FlashStartAdd + 0x2AAA) = 0x55;
   *(FlashStartAdd + 0x5555) = 0xA0;
   *(FlashStartAdd + AddStart + i) = *(&WriteData[0] + DataStart + i);
  }
}
```

10 ms 周期保存数据测试程序代码如下：

```
void StorageData(void)
{
  if(WriteTimer1 >= 10)//定时 10 ms  全局变量
  {
   WriteTimer1 = 0;
   WriteData[0] =0x1111;
   WriteData[1] =0x2222;
   WriteData[2] =0x3333;
   WriteData[3] =0x4444;
   WriteData[4] =0x5555;
   WriteData[5] =0x6666;
   FlashWrite(0,WriteRepeat,6);
   if(WriteRepeat < 0xFFFFF)//未写满时有效
   {
    WriteRepeat=WriteRepeat+6;
   }
  }
}
```

如图 4 所示，操擦完成后对 Flash 进行写操作测试，测试结果可查看寄存器值。

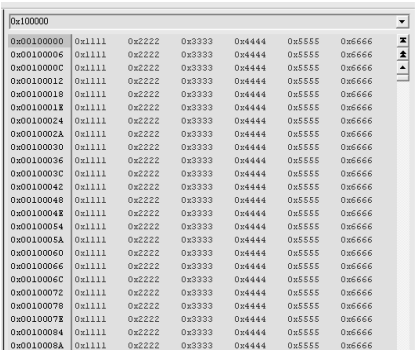

图 4 写操作后 Flash 中数据

2.4 读操作程序设计

与擦除操作和写相比，Flash 读操作较简单，可直接将某一地址中的数据取出。笔者设计的测试实例如下：以 10 ms 为周期，每周期读取 6 个字，将上述写入的数据读出并通过串口发送，最终由串口助手显示。

Flash 读操作程序代码如下：

```
void FlashRead(Uint32 DataStart, Uint32 AddStart, Uint16 Length)  // 目的地址，源地址，地址长度
{
  Uint32 i=0;
  Uint16 temp=0;
  WP = 0;//读操作时为低电平，禁止擦、写
  For (i = 0; i < Length; i++)
  {
  Temp  =  *(FlashStartAdd + AddStart + i);
  *(&ReadData[0] + DataStart + i)  =  temp;
  }
}
```

10 ms 周期读取数据测试程序代码如下：

```
void TakeData(void)
{
  if(ReadTimer1 >= 10)//定时 10ms  全局变量
  {
   ReadTimer1 = 0;
   FlashRead(0, ReadRepeat, 6);
   if(ReadRepeat < 0xFFFFF)
   {
    ReadRepeat=ReadRepeat+6;
   }
  }
}
```

写操作完成后对 Flash 进行读操作测试，测试结果通过串口助手查看如图 5 所示。

图 5 串口读取 Flash 中数据

3 结束语

测试实例结果表明：该方法具有一定的实用价值，充分利用 TMS320F28335 与片外 Flash 存储器之间的数据总线和地址总线，快速地实现 Flash 的

读、写、擦除操作，不仅实现了发动机试车数据的存储，而且可随时读取/擦除单次试车数据或指定地址的试车数据，对发动机关键数据的保存与分析具有较为重要的意义。

参考文献：

[1] 刘陵顺, 高艳丽, 张树团, 等. TMS320F28335 DSP 原理及开发编程[M]. 北京: 北京航空航天大学出版社, 2011: 103-118.

[2] 顾卫钢. 手把手教你学 DSP—基于 TMS320X281x[M]. 北京: 北京航空航天大学出版社, 2011: 127-148.

[3] 周世新. 大容量 Flash 与 DSP 接口技术的实现[J]. 无线电工程, 2006, 6(3): 57-58.

[4] 李娟, 王金海, 王敏, 等. DSP 外挂 FLASH 在线编程的研究与实现[J]. EIC, 2009(1): 114-116.

[5] 苏奎峰, 吕强, 常天庆, 等. TMS320X281x DSP 原理及 C 程序开发[M]. 北京: 北京航空航天大学出版社, 2008: 367-375.

****

（上接第 14 页）

图 2　系统生成 Hash 函数界面

图 3　数据完整性验证界面

4　结束语

笔者提出一种基于 Hash 函数的数据完整性验证方案，基于改进的 SM3 算法，设计、实现了导弹数据数字化登记系统。功能试用结果表明：该系统能够完成数据完整性验证，并将验证结果告知用户。下一步，可将数字签名技术应用于该系统，在验证数据完整性的基础上明确数据来源，防止可能出现的代签名导致责任人不明、数据来源不明确等问题。

参考文献：

[1] 赵军, 曾学文, 郭志川. 国产与国外常用杂凑算法的比较分析[J]. 网络新媒体技术, 2018, 7(5): 58-62.

[2] 徐津, 温巧燕, 王大印. 一种基于 Hash 函数和分组密码的消息认证码[J]. 计算机学报, 2015, 38(4): 793-803.

[3] 王小云, 于红波. 密码杂凑算法综述[J]. 信息安全研究, 2015, 1(1): 19-30.

[4] WANG X Y, YU H B. How to break MD5 and other hash functions[C]//Advances in Cryptology EUROCRYPT 2005. Heidelberg: Springer Berlin, 2005: 19-35.

[5] 朱宁龙, 戴紫彬, 张立朝, 等. SM3 及 SHA-2 系列算法硬件可重构设计与实现[J]. 微电子学, 2015, 45(6): 777-780, 784.

[6] 王小云, 于红波. SM3 密码杂凑算法[J]. 信息安全研究, 2016, 2(11): 983-994.

[7] 国家密码管理局. SM3 密码杂凑算法: GM/T0004—2012 [S]. 北京: 国家密码管理局, 2012.

[8] 郭小威, 向哲. 基于蚁群算法的舰载武器共架发射时域协调方法[J]. 兵工自动化, 2019, 38(10): 61-65.