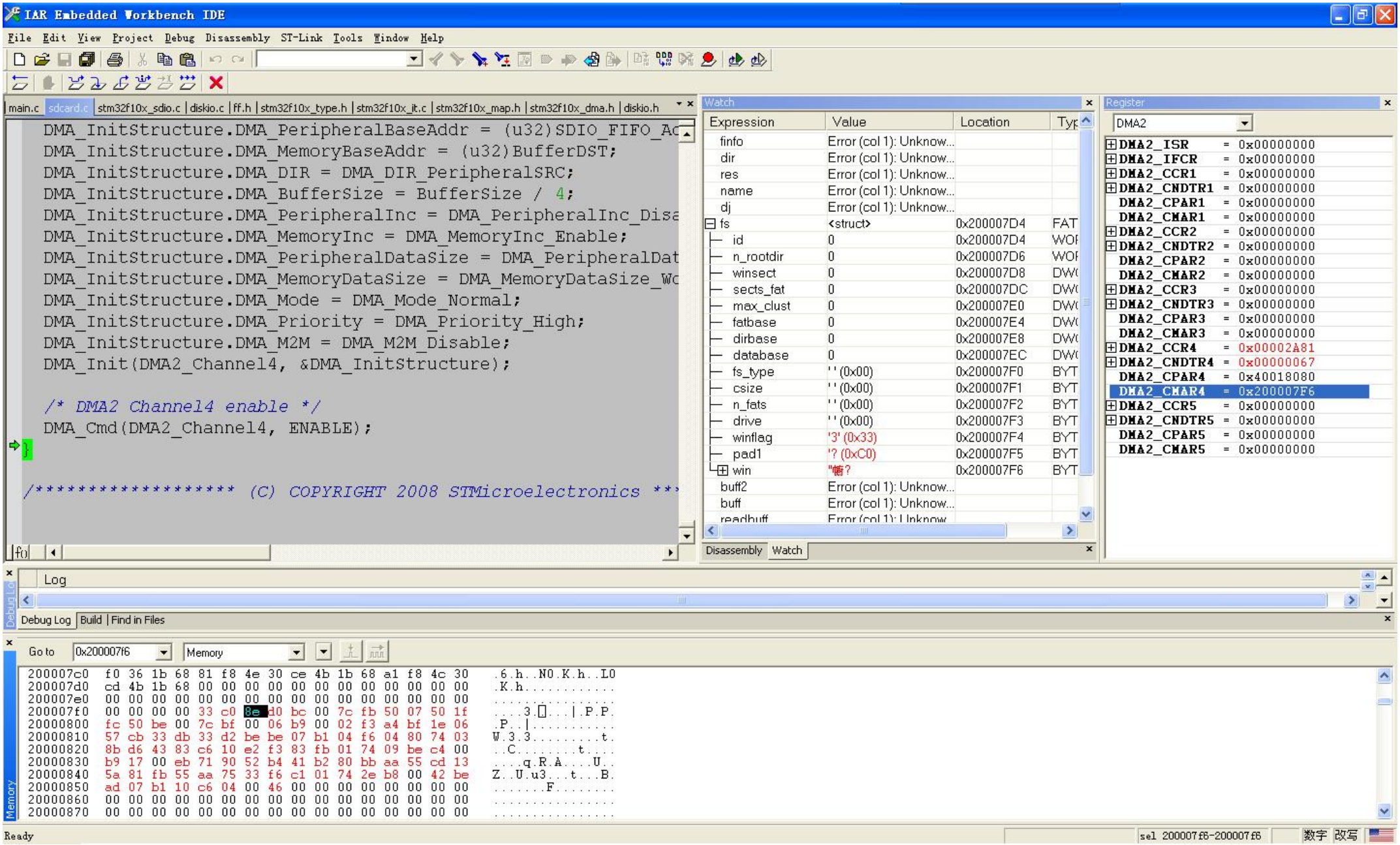
这两天调试 stm32 单片机的 sdio 接口读写 sd 卡，把调试经验总结一下。

我首先采用的是官方例程里面得 dma 方式，直接使用官方提供的 sdcard.c 文件。首先声明一个 uint32 类型的数据缓冲区，读写 sd 卡的每个扇区完全正常，随后添加 fatfs 的文件，然后改好接口函数，其中有一处做了强制转换，就是 fatfs 声明的缓冲区都是 uint8*类型的，而 sdcard.c 文件要求为 uint32*类型，所以我把 uint8*强制转换为 uint32*类型。编译成功后执行，结果 fatfs 的初始化函数返回“没有文件系统”，经过多次小改动后重试，问题依然如此，后来我怀疑是我强制转换出的问题，为了验证我的想法，我做了如下程序：

```
#include <stdio.h>
#include <string.h>

void main(void)
{
	unsigned char i;//AA[] = {0,1,2,3,4,5,6,7,8,9,0,4};
	unsigned char * aa;
	unsigned int * bb;

	for(i=0;i<10;i++)
	{
		*(aa+i) = i;
	}

	bb = (unsigned int *)aa;
	printf("bb is: 0X%04x \n",*bb);
	bb = (unsigned int *)(aa+1);
	printf("bb is: 0X%04x \n",*bb);
	while(1);
}
```

以上程序验证结果为：在 IAR EWARM 5.40 中，强制转换不会对数据存取产生影响。

既然不是这个问题，那会是什么呢？！我只好单步执行程序，一步步追踪下去，看看到底问题出在哪里。功夫不负有心人，最后终于发现出问题的地方，如下图:


IAR Embedded Workbench IDE
File Edit View Project Debug Disassembly ST-Link Tools Window Help
main.c sdcard.c stm32f10x_sdio.c diskio.c ff.h stm32f10x_type.h stm32f10x_it.c stm32f10x_map.h stm32f10x_dma.h diskio.h
DMA_InitStructure.DMA_PeripheralBaseAddr = (u32)SDIO_FIFO_Ac
DMA_InitStructure.DMA_MemoryBaseAddr = (u32)BufferDST;
DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralSRC;
DMA_InitStructure.DMA_BufferSize = BufferSize / 4;
DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disa
DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable;
DMA_InitStructure.DMA_PeripheralDataSize = DMA_PeripheralDat
DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_Wo
DMA_InitStructure.DMA_Mode = DMA_Mode_Normal;
DMA_InitStructure.DMA_Priority = DMA_Priority_High;
DMA_InitStructure.DMA_M2M = DMA_M2M_Disable;
DMA_Init(DMA2_Channel4, &DMA_InitStructure);
/* DMA2 Channel4 enable */
DMA_Cmd(DMA2_Channel4, ENABLE);
}
/******************* (C) COPYRIGHT 2008 STMicroelectronics ***
Watch
Expression Value Location Typ
finfo Error (col 1): Unknow...
dir Error (col 1): Unknow...
res Error (col 1): Unknow...
name Error (col 1): Unknow...
dj Error (col 1): Unknow...
fs <struct> 0x200007D4 FAT
id 0 0x200007D4 WOF
n_rootdir 0 0x200007D6 WOF
winsect 0 0x200007D8 DW(
sects_fat 0 0x200007DC DW(
max_clust 0 0x200007E0 DW(
fatbase 0 0x200007E4 DW(
dirbase 0 0x200007E8 DW(
database 0 0x200007EC DW(
fs_type '' (0x00) 0x200007F0 BYT
csize '' (0x00) 0x200007F1 BYT
n_fats '' (0x00) 0x200007F2 BYT
drive '' (0x00) 0x200007F3 BYT
winflag '3' (0x33) 0x200007F4 BYT
pad1 '? (0xC0) 0x200007F5 BYT
win "慚? 0x200007F6 BYT
buff2 Error (col 1): Unknow...
buff Error (col 1): Unknow...
Disassembly Watch
Register
DMA2
DMA2_ISR = 0x00000000
DMA2_IFCR = 0x00000000
DMA2_CCR1 = 0x00000000
DMA2_CNDTR1 = 0x00000000
DMA2_CPAR1 = 0x00000000
DMA2_CMAR1 = 0x00000000
DMA2_CCR2 = 0x00000000
DMA2_CNDTR2 = 0x00000000
DMA2_CPAR2 = 0x00000000
DMA2_CMAR2 = 0x00000000
DMA2_CCR3 = 0x00000000
DMA2_CNDTR3 = 0x00000000
DMA2_CPAR3 = 0x00000000
DMA2_CMAR3 = 0x00000000
DMA2_CCR4 = 0x00002A81
DMA2_CNDTR4 = 0x00000067
DMA2_CPAR4 = 0x40018080
DMA2_CMAR4 = 0x200007F6
DMA2_CCR5 = 0x00000000
DMA2_CNDTR5 = 0x00000000
DMA2_CPAR5 = 0x00000000
DMA2_CMAR5 = 0x00000000
Log
Debug Log Build Find in Files
Go to 0x200007f6 Memory
200007c0 f0 36 1b 68 81 f8 4e 30 ce 4b 1b 68 a1 f8 4c 30 .6.h..N0.K.h..L0
200007d0 cd 4b 1b 68 00 00 00 00 00 00 00 00 00 00 00 00 .K.h............
200007e0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
200007f0 00 00 00 00 33 c0 8e d0 bc 00 7c fb 50 07 50 1f ....3.....|.P.P.
20000800 fc 50 be 00 7c bf 00 06 b9 00 02 f3 a4 bf 1e 06 .P..|...........
20000810 57 cb 33 db 33 d2 be be 07 b1 04 f6 04 80 74 03 W.3.3.........t.
20000820 8b d6 43 83 c6 10 e2 f3 83 fb 01 74 09 be c4 00 ..C........t....
20000830 b9 17 00 eb 71 90 52 b4 41 b2 80 bb aa 55 cd 13 ....q.R.A....U..
20000840 5a 81 fb 55 aa 75 33 f6 c1 01 74 2e b8 00 42 be Z..U.u3...t...B.
20000850 ad 07 b1 10 c6 04 00 46 00 00 00 00 00 00 00 00 ....F........
20000860 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
20000870 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
Memory
Ready
sel 200007f6-200007f6
数字 改写

原来是 DMA 传输出了错误，fatfs 声明的 uint8*的缓冲区不是从整 4 倍字节地址开始的，而 DMA 控制器在操作 uint32*类型地址的时候必须从整 4 倍字节地址开始，由于强制转换，产生了 2 个字节的偏移，从而导致最终的数据错误。

既然发现问题所在，就想办法解决，我在 fatfs 和 sdcard 的接口函数处又声明了一个 uint32*类型的缓冲区，每次读写扇区都通过这个缓冲区转换一下，结果试验成功，fatfs 终于能操作 sd 卡了。

接下来一个问题是，每个扇区都这样操作，必然大大影响 sd 卡的读写速度，还得继续想办法解决这个问题。随后我想到，既然 DMA 出问题，我就不用它了，我用中断方式吧。这个方式切换很好办，官方例程都做好了，只需初始化的时候选一下就行了。把程序改好后运行---不好用！仔细检查程序，没发现问题，只好继续繁琐的单步调试。这次发现，出错原因是由于 SDIO 读函数返回“SD_RX_OVERRUN”造成的，从程序中看不出所以然来，只好问百度 google 两位大神了。网上查询得知，原来许多人在用 stm32 的 sdio 时，遇到此问题，高人指明，将操作 sd 卡的 4 位总线模式改为 1 位就行了，我改完一试，果然好用了。随后经过分析 datasheet 和上网查询，得出 4 位总线模式不好用的原因：由于 sd 卡 4 位总线模式读写太快，单片机中断函数来不及取走 fifo 里面的数据，导致数据溢出出错，通过降低 sd 卡时钟频率的方式也能避免出错。

调试总结，首先说 fatfs，这个文件系统主要是针对小型单片机而编写的，对字节操作很便捷，但是对 32 整型地址操作支持不是很好，导致上面说的 DMA 操作出问题。如果想用 DMA 操作，需要使用支持以上操作的文件系统。我在附件例程里面大面积注释掉的部分，是一个叫 “DOSFTAT”的文件系统，该文件系统就支持 DMA 操作，只是接口函数不如 fatfs 的友好。

其次说一下读写速度，由于我这的应用对速度基本没什么要求，所以我没有详细测试，根据网友测试结果，如果采用 DMA+4 位总线，在 24MHz 时钟下，据说读取速度能达到 10MB/秒，这个已经很快了。不过对于速度不高的应用，我还是建议采用 1 位总线+中断模式，这样即兼容性好又能保证稳定性，还省点电。

最后说明一下我提供的例程，该工程在 IAR EWARM 5.40 下编译成功，主要实现功能是用 fatfs 从 sd 卡读取一个 txt 文件，并通过串口把文件名和内容打印出来。注释掉的部分是我从一个外国网站上下的 dofat 的文件系统，感兴趣的朋友可以试一下。

此文是我调试过程中进行的总结，肯定存在一些错误，欢迎指正。

oet

2009-10-24