

Ling.Ju

第六讲 川流不息

百川东到海,何时复西归!

本讲学习目标:

1、将 STM32 进阶学习。

A: STM32 的 FLASH

FLASH 对嵌入式业者并不陌生,程序数据可以被存储在里面,有人会问:已经学会将程序通过编程器或仿真器烧写 FLASH,为何今天还要去讲如何编程 FLASH?

其实这部分内容是希望大家能对 FLASH 有所了解,会编程 FLASH 之后,就可以创建自己的 BootLoad、以及在线更新产品的代码等。(有些地方称之为 IAP 编程)

需要注意的是:

- 1.FLASH 有页的概念,在 STM32 中 FLASH 根据不同的芯片容量,有 1K、2K 一页之分。
 - 2.FLASH 在读写前后需要,解锁和加锁。
 - 3.写 0 容易,写 1 难(STM32 中由数据 1 改写为 0 很方便,但是若 0 需要恢复成 1 需要有页擦出)



Ling.Ju

库函数名	功能		
FLASH_Unlock	解锁 FLASH	无参数	
FLASH_Lock	加锁 FLASH	无参数	
FLASH_ProgramWord	写入字	地址、数据	
FLASH_ErasePage	页擦除	页地址	

编程实例:

#define FLASH_adr 0x08008000 //要写入数据的地址

#define FLASH_dat 0x65368239 //要写入的数据

//读数据

u32 tmp; //定义缓冲

tmp=*(vu32*)(FLASH_adr); //读取地址中的数据

// 注意这个指针的读法

// 1. vu32* 代表指向指针的类型

// 2. (vu32*)(FLASH_ADR) 代表地址

// 3. *(vu32*)(FLASH_ADR) 取这个地址的中数据

//写数据

FLASH_Unlock();

FLASH_ProgramWord(FLASH_adr,FLASH_dat);

FLASH_Lock();



Ling.Ju

//擦除数据

FLASH_Unlock();

FLASH_ErasePage(FLASH_adr);

FLASH_Lock();

B: STM32的ID

ID 是什么?其实就是产品的电子签名。为了提高芯片的加密能力,防止反向工程的出现,经常会使用到这种唯一性的身份标示,刚才在 FLASH 中已经提及数据在 FLASH 中的读取,下面我们就来看看 STM32 的 96 位数据。

基地址: 0x1FFF F7E8	
地址偏移: 0 x00 16 位	地址偏移: 0x02 16 位
地址偏移: 0 x04 32 位	地址偏移: 0 x08 32 位

static u32 CpuID;

CpuID=*(vu32*)(0x1FFFF7E8); // 如果读不懂这个指针,请看上节中的注释

注意: 只可读不可写。

Ling.Ju

C: STM32 的 "旺财"

看门狗是干啥的?防止你 CPU 意外死掉的。狗狗在开启后,过一段时间喂一下,否则 狗狗都死掉了的~~~~(>_<)~~~。

STM32 的看门狗有两种暂时先介绍"独立看门狗"。

独立看门狗(IWDG)由专用的低速时钟(LSI)驱动,即使主时钟发生故障它也仍然有效。

IWDG 最适合应用于那些需要看门狗作为一个在主程序之外,能够完全独立工作,并且对时间精度要求较低的场合。

看清楚一下这段话:

IWDG主要性能

- 自由运行的递减计数器
- 时钟由独立的RC振荡器提供(可在停止和待机模式下工作)
- 看门狗被激活后,则在计数器计数至0x000时产生复位

这让我们知道了,时钟是哪来的,计数器是怎么工作的。

这个是喂狗的时间控制。如果超过这个时间狗狗就要按下重启开关了。

看门狗超时时间(40kHz的输入时钟(LSI))

预分频系数	PR[2:0]位	最短时间(ms) RL[11:0] = 0x000	最长时间(ms) RL[11:0] = 0xFFF		
/4	0	0.1	409.6		
/8	1	0.2	819.2		
/16	2	0.4	1638.4		
/32	3	0.8	3276.8		
/64	4	1.6	6553.6		
/128	5	3.2	13107.2		
/256	(6或7)	6.4	26214.4		

注: 这些时间是按照40kHz时钟给出。实际上,MCU内部的RC频率会在30kHz到60kHz之间变化。 此外,即使RC振荡器的频率是精确的,确切的时序仍然依赖于APB接口时钟与RC振荡器时钟 之间的相位差,因此总会有一个完整的RC周期是不确定的。

看门狗功能处于VDD供电区,即在停机和特机模式时仍能正常工作。



接下来看下白菜叔叔实例:

//肉包子函数

IWDG_ReloadCounter(); //肉包子喂狗

//唤醒狗狗的函数 void IWDG_Configuration(void) { /* 写入 0x5555,用于允许狗狗寄存器写入功能 */ IWDG_WriteAccessCmd(IWDG_WriteAccess_Enable); /* 狗狗时钟分频,40K/256=156HZ(6.4ms)*/ IWDG_SetPrescaler(IWDG_Prescaler_256); /* 喂狗时间 Xs/6.4MS=XXX .注意不能大于 0xfff*/ IWDG_SetReload(XXX); /* 喂狗*/ IWDG_ReloadCounter(); /* 使能狗狗*/ // IWDG_Enable(); }

D: STM32的 "printf"

哈哈" HELLO C!"。

这句简单的话时候,是否想起了初出茅庐的我们?

在 STM32 中确实也可以使用 printf 语句,只是这是个单片机,并不是 PC 机,如何才能让他知道输出到哪里呢?

这就牵涉到了一个输出定向的问题了。什么叫定向?就是告诉 CPU 你的方向。你指向哪里。

第一步,我们要知道一个勾:

Options fo		'Target 1' Output List	ing User	C/C++ A	.sm I	inker	Debug Vtil:	ities	X
		STM32F103VI			Code G	ieneratior	1		
Operatii	ng system:	None		¥	V U	se MicroL	Module Optimiza IB Γ ime Code Gener	Big Endian]
⊢Read/0	Only Memo	ory Areas			⊢Read∧	Write Mer	nory Areas		
default	off-chip	Start	Size	Startup	default	off-chip	Start	Size	NoInit
	ROM1:			0		BAM1:			
	ROM2:			0		RAM2:			
	ROM3:			0		RAM3:			
	on-chip	1				on-chip	1)]	
▽	IROM1:	0x8000000	0x40000	•	V	IBAM1:	0x20000000	0xC000	
	IROM2:			0		IRAM2:			
			OK	Can	cel	De	faults		Help

这是 MDK 附带的一个 LIB, 会自动调用和添加, 仅需打钩而已。

第二步, 我们要添加熟悉的头文件

#include <stdio.h>



Ling.Ju

第三步也是关键的一步,是在设置好串口过后,你需要添加定向用的文件,也就是 printf 需要调用的一个文件。

```
/*定义 fpute 此函数为 printf 所用*/
int fputc(int ch)

{

USART_SendData(USART1, (u8) ch);

/* Loop until the end of transmission */
while(USART_GetFlagStatus(USART1, USART_FLAG_TXE) == RESET);
return ch;
}

这三部分过后,呵呵,就可以使用 printf 啦!
printf("无符号十进制形式:%u\r\n",X);
printf("十六进制形式:%x\r\n",X);
```

注意:一定要注意哦,这些事之外千万不要忘记按照之前讲义上去配置 USART 哦,否则就没用了的。



Ling.Ju

E: 课后结语

第六讲旨在学习 STM32 一些杂七杂八的内部资源。之前的 5 讲若是听过的话,那么这些都能很快速的上手,到此稀里糊涂学 STM32 讲义已经完成了 GPIO USART EXIT NVIC TIMx RTC SysTick ID IWDG 这些部分的指导,还有 SPI、PWM、WWDG 将在以后的课程中继续进行,原本是想本节课讲有关 SPI 的内容,实在来不及去完成,故在此对读者们深表歉意。

至于 FATFS, 我会在讲完 SPI 以后将 FATFS 的移植移入讲义的安排。敬请关注~

本人最近忙于新唐 M0 的学习与讲座,并且担任了 21 ic 的一个小斑竹,进行新唐 M0 的学习指导与文档规整的工作,故稀里糊涂学 STM32 进度有些缓慢,加之清明扫墓等原因,实在对不起各位看官。在此深表歉意。不过我会努力吧稀里糊涂学 STM32 继续的~哈哈~

D: ሣ	T课笔记		