SAM4N 工程模板搭建

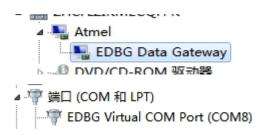
SAM4N 学习笔记

版本号: 1.00

日期: 2013/10/25

一、需要安装的软件:

因为笔者是使用 MDK-ARM 开发的版本是 4.72,所以需要安装这个工具,具体在哪里下载自行放狗或问度娘。除了这个重要工具以为,还需要安装 Atmel 官方的 Atmel Studio,为什么安装这个软件呢?其实不安装也是可以的,但是不安装需要自备一条串口线,板上自带的 USB CDC 就不能用了,因为找不到驱动。安装 Atmel Studio 以后,会自动安装 USB 驱动,这事连接板子 PC 就识别并安装驱动了,安装好后在"设备管理器"中会看到枚举出如下两个设备:



这里的 COM8 是自己 PC 枚举的,不同的 PC 得出的端口号会不同。如果有这两个,说明你的板子已经可以用了,如果只有前面一个,那也是可以下载程序的。

二、创建工程模板:

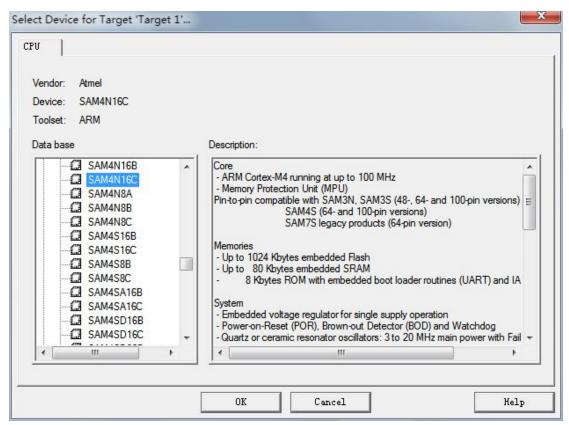
关于软件的安装我就不多说了,都是一直点下一步就 OK 的。接下来开始搭建我们的第一个工程,也就是所说的"工程模板"。下去我都是以这个为模板进行其他例程的开发。首先新建一个文件夹,命名为"工程模板",打开文件夹,在里面依次建立

"Applications"、"Drivers"、"CMSIS"、"Listing"和"build"五个文件夹,如下图:

名称	修改日期	类型	
Applications	2013/10/25 18:20	文件夹	
📗 build	2013/10/25 18:20	文件夹	
CMSIS	2013/10/25 18:20	文件夹	
Drivers	2013/10/25 18:20	文件夹	
listing	2013/10/25 18:20	文件夹	

简要说明一下他们的用途,Applications 文件是用来存放用户应用程序的; Drivers 用来存放开发的各种驱动; CMSIS 存放芯片相关的文件,如启动文件等; build 主要存放编译输出的一些文件; Listing 用于存放链接输出的文件。

建立好这些文件夹以后,我们开始建立工程。启动 MDK 软件,点击菜单"Project"菜单下的"New uVision Project"新建一个项目,保存到工程模板文件夹下,项目名称为"Project",选择 CPU 为 Atmel 公司的 SAM4N16C,如下图:



点击 "OK"完成工程创建。接下来要对工程进行一些配置,首先要把需要的芯片相关的头文件和启动文件复制到刚刚建立的 "CMSIS"文件夹中。到哪里去复制芯片相关头文件好启动文件呢?这是很多人比较疑惑的事情。其实有两个方法可以拿到这些文件,第一种方法是到 Atmel 提供的软件包里面去找,还有一种是到 MDK 的安装目录里面去找。这里我们选择后者。

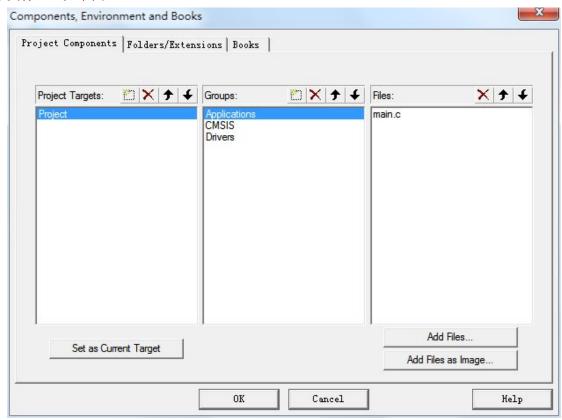
打开 MDK 的安装目录,在"\ARM\INC\Atmel\SAM4N"下面就是需要的头文件,如下图:

	修改日期	类型	大小
component	2013/9/25 15:34	文件夹	<u>_</u>
instance	2013/9/25 15:34	文件夹	
🃗 pio	2013/9/25 15:34	文件夹	
sam4n.h	2013/6/24 10:35	H 文件	2 KB
sam4n8a.h	2013/6/24 10:35	H 文件	22 KB
sam4n8b.h	2013/6/24 10:35	H文件	24 KB
sam4n8c.h	2013/6/24 10:35	H 文件	25 KB
sam4n16b.h	2013/6/24 10:35	H 文件	24 KB
sam4n16c.h	2013/6/24 10:35	H 文件	25 KB
system_sam4n.h	2013/6/24 10:35	H 文件	4 KB

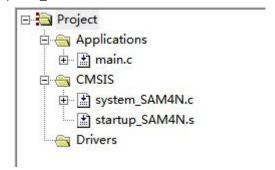
在我们的工程中的 CMSIS 文件夹下建立一个 "include" 子文件夹,并将这些头文件复制到该文件夹下;接下来是启动文件,打开 "\ARM\Startup\Atmel\SAM4N",有两个文件,一

个是启动汇编,一个是 C 文件,里面有 SAM4N 启动时配置的一些操作,如时钟配置等。将这两个文件复制到 "CMSIS"文件夹下的 Source 文件夹。也许有人会问,为什么需要复制这些文件到工程中呢?这些是芯片必须的文件,其实不复制过来 MDK 也会自己去索引这些文件,但是这样做可以避免工程被拷贝到其他没有安装 MDK 或 MDK 版本低的地方还可以找到这些文件。

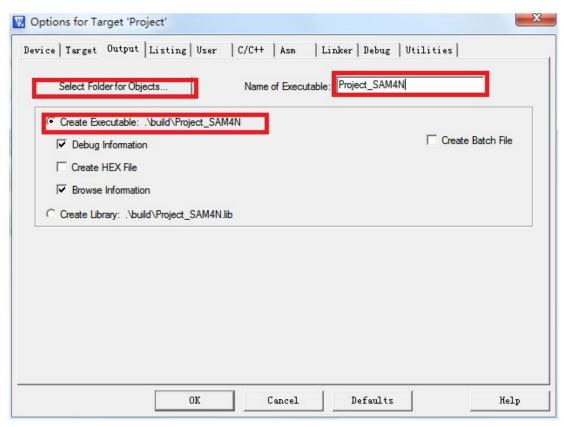
接下来要配置一下 MDK 工程的文件组,分别建立"Applications"、"CMSIS"和"Drivers" 三个文件组,如下图:



同时给"Applications"添加一个新文件 main.c,然后将"startup_SAM4N.s"和"system_SAM4N.c"文件添加到"CMSIS"文件组下,如下图:



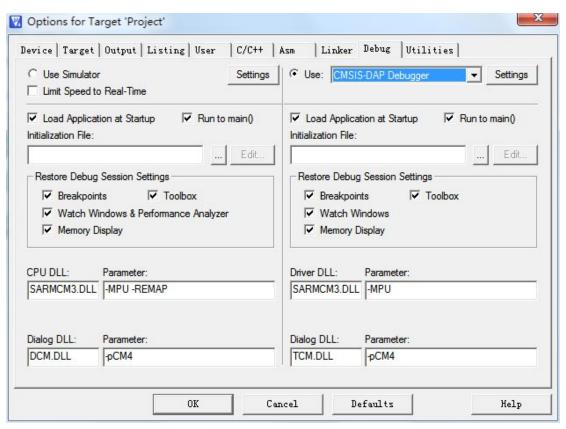
打开 "Target Options",选择输出文本到"output"标签,选择输出文件到"build"文件夹如下图:



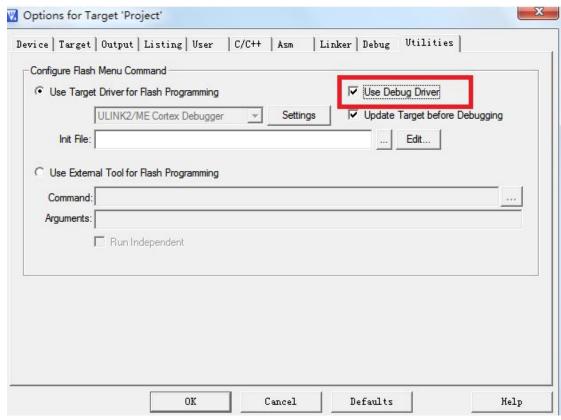
点击 "Listing" 标签,设置输出到 "Listing" 文件夹。点击 "C/C++" 标签,在"Define" 中添加"__SAM4N16C__"宏定义,并在"Include Paths"中添加"Applications"、"CMSIS\include" 和"Drivers" 三个目录,如下图:

ice Target Output Listin	g User C/C++ Asm Linker Debug	Utilities
Preprocessor Symbols		
Define: SAM4N16C_		
Undefine:		
Language / Code Generation	664	Warnings:
	Strict ANSI C	
Optimization: Level 0 (-00)	Enum Container always int	<unspecified> ▼</unspecified>
Optimize for Time	Plain Char is Signed	Thumb Mode
Split Load and Store Multiple	Read-Only Position Independent	☐ No Auto Includes
One ELF Section per Function	Read-Write Position Independent	
Include Paths Misc	S\include;.\Drivers	
Controls Compiler -c -cpu Cortex-M4 -D	_MICROLIB -g -00apcs=interworkasminterlea	veasm_dir ".\\Listing\\" - 🔺
control I.\Applications -I.\CM:	SIS\include -I.\Drivers	+

点击"Debug"标签,选择使用"CMSIS-DAP Debugger"仿真器仿真,如下图:



点击"Utilities"标签,勾选使用 Debug Driver 进行 Flash 编程,如下图。



到此,新建的工程算是配置完成,接下来就是小试牛刀的时候了,打开 main.c, 开始代码的编写,看了一下数据手册,随便写了个代码测试,如下:

#include "sam4n.h"

```
#define LED_PIN
   void delay(uint32_t x)
   {
       uint32_t a, b;
       for (a = 0; a < 5000; a++)
           for (b = 0; b < x; b++);
       }
   }
   int main(void)
   {
       /*使能 LED 管脚*/
       PIOB->PIO_PER |= (0x01 << LED_PIN);
       /*使能 LED 管脚输出*/
       PIOB->PIO_OER |= (0x01 << LED_PIN);
       while (1)
       {
           /*输出低电平,点亮 LED*/
           PIOB->PIO_CODR = (0x01 << LED_PIN);
           /*延时*/
           delay(300);
           /*输出高电平,关闭 LED*/
           PIOB->PIO_SODR = (0x01 << LED_PIN);
           /*延时*/
           delay(300);
       }
    点击编译,点击工具栏上的"Download"下载到板子,果然灯闪动起来了,还真是误
打误撞,太 easy 了。
```