ARC

ARC 平台简介

支持	www.armrunc.com http://armrunc.taobao.com

Change History

版本号	更改描述	作者	修改时间
1.0.0	第一版	armrunc	2012-03-05

目录

第2章 ARC 平台

2.1 ARC 平台简介

本平台的主芯片为 ST 的 STM32F103RBT6.
硬件设计充分考虑了实验的方便性.
软件设计严格遵守 driver/middleware/application 分层结构，所有的寄存器操作都会在 driver 层实现（ST 3.5 驱动），middleware 实现具体的模块功能，application 层实现实例的功能，这种严格的结构化代码使得代码非常容易移植，易懂， 并且所有的层的源代码全部开放。
软件注释使用 doxygen 脚本注释。

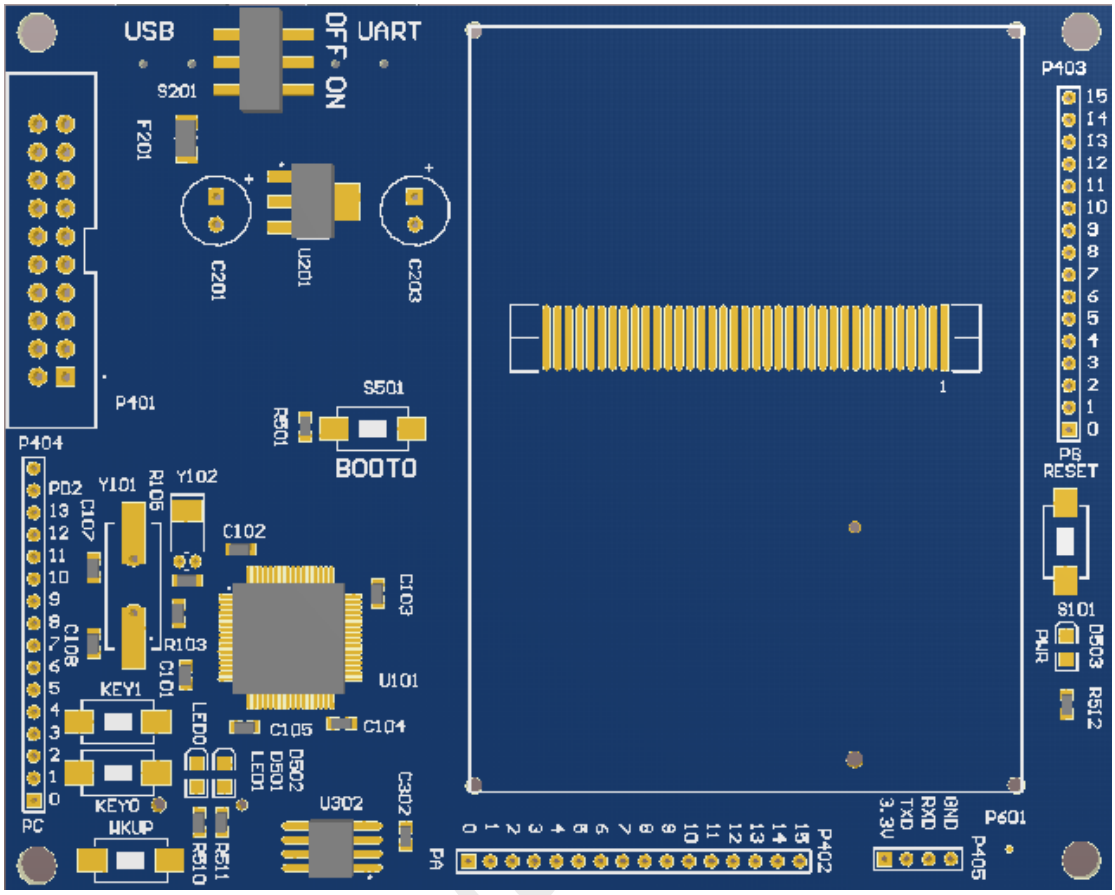
2.2 ARC 平台硬件

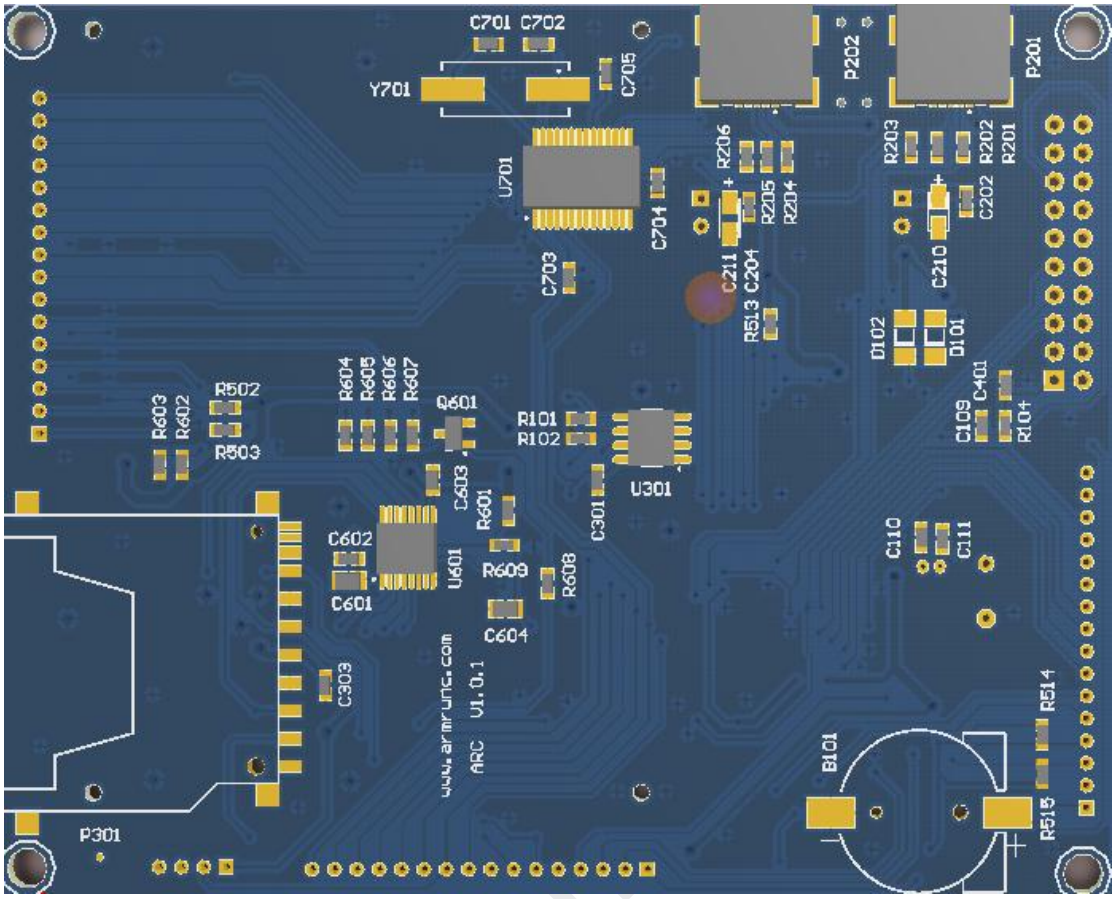
该平台硬件特点：

- 三个 LED，其中一个为电源指示灯，另外两个用来做跑马灯和其它指示；
- 三个按键，其中一个是 wake up 中断，另外两个用于外部中断输入和 ADC 检测输入；
- EEPROM 用于 I2C 实验；
- SPI FLASH 用于 SPI 接口实验和存储数据；
- SD 卡接口，用于 SPI SD 卡操作；
- LCD 模块用于显示；
- 触摸屏控制用于屏幕交互式操作；
- 平台上集成了 PL2303 用于 USB 转串口，只需要一个 USB 线就能调试；
- JTAG 接口用于在线调试；
- 所有的 GPIO 引出，便于扩展实验；
- 两个 USB 接口，其中一个作为串口调试口，另外用于 STM32 USB 实验，同时这两个 USB 口还用来供电；
- RTC 实时时钟，带电池；
- 电源开关；
- I2C 使用 STM32 的 I2C 接口，在软件实现上，你既可以用 GPIO 模拟 I2C 也可以使用 STM32 的硬件 I2C 来实现 I2C 协议，ARC 平台使用 STM32 的硬件 I2C 实现；
- 所有的 SPI 器件全部连接到 SPI 硬件接口，软件实现方法可以用 GPIO 模拟 SPI 或者使用 STM32 的硬件 SPI 实现 SPI 协议；
- LED 和 按键的设计充分考虑到应用实例，分别连接到 TIM 和 ADC，无需飞线，方便实现 PWM 和 ADC 采样实验；

该平台的硬件电路图如下：

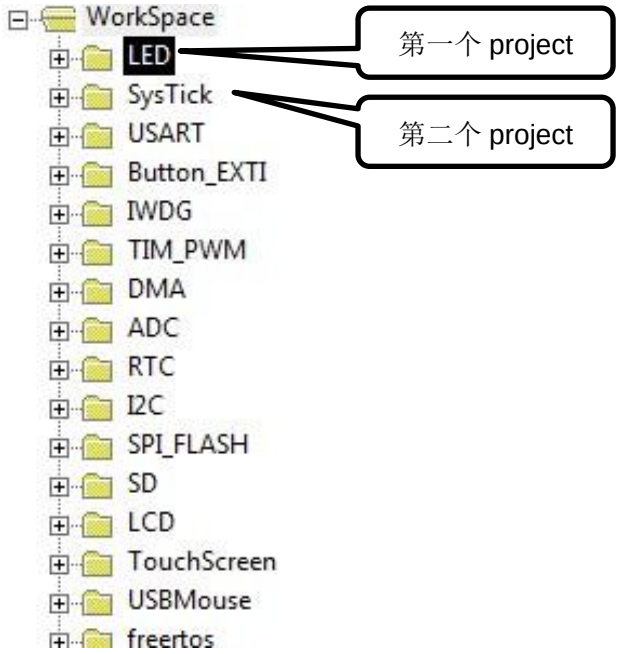
其中 S501 用来选择启动方式， S501 短接时（按下时），从串口烧录数据，S501 断开时，正常从内部 flash 启动。





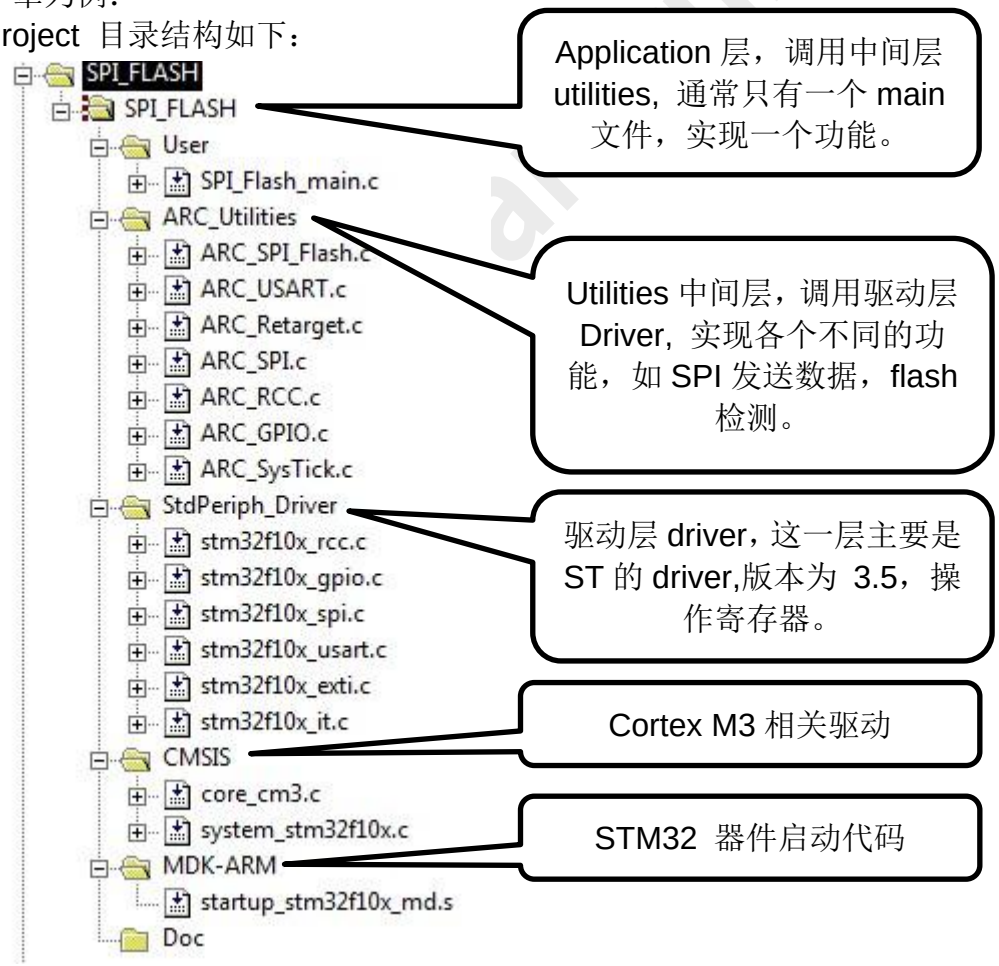
2.3 ARC 平台软件

ARC 平台使用的编译器为 keil MDK 4.23，代码编辑器建议使用 source insight。所有的 keil project 都按实例的顺序包含到 keil workspace。 这样，你只要打开一个 workspace，就能按讲解顺序编译开发所有的 project，目录结构如下：
Workspace 目录结构如下：



在每个 project，文件名和文件摆放位置也是严格按照功能摆放，以 SPI flash 一章为例：

Project 目录结构如下：



C 文件的构造如下：

Main 文件 SPI_FLASH_main.c：

SPI_Flash_main.c

```
37
38  /**
39    * @brief  Main program, SPI flash write/read example.
40    * @param  None
41    * @retval None
42    */
43  int main(void)
44  {
45      uint8_t  Tx_Buffer[] = "ARC SPI FLASH Example";
46      uint8_t  Rx_Buffer[512] = "flash not present or not recognized\n";
47      uint16_t BufferSize = sizeof(Tx_Buffer) / sizeof(*Tx_Buffer);
48      uint32_t FlashAddr = 0x0;
49
50      /* initialize SysTick for exact time de
51      ARC_SysTick_Init();
52
53      /* initialize UART */
54      ARC_COM_Init();
55      USART_Cmd(USART1, ENABLE);
56
57      /* initialize UART */
58      ARC_SPI_Flash_Init();
59      SPI_Cmd(SPI1, ENABLE);
60
61      /* check if the flash recognized */
62      ARC_SPI_FLASH_ID_check();
63
64      if (spi_flash_found)
65      {
66          /* Erase SPI FLASH Sector to write on */
67          ARC_FLASH_EraseSector(FlashAddr);
68
69          /* Write Tx_Buffer data to SPI FLASH memory */
70          ARC_FLASH_WriteBuffer(Tx_Buffer, FlashAddr, BufferSize);
71
72          /* Read data from SPI FLASH memory */
73          ARC_FLASH_ReadBuffer(Rx_Buffer, FlashAddr, BufferSize);
74      }
75
76      while (1)
77      {
78          printf("Tx: %s\n", Tx_Buffer);
79          printf("Rx: %s\n", Rx_Buffer);
80
81          /* delay one second */
82          ARC_SysTick_Delay(1000);
83      }
84  }
85
```

函数注释

调用 Utilities 层的 API，延迟初始化都会集中在这个函数内，调用完这个函数就能使用延迟函数了

SPI flash 初始化并使能。初始化 SPI 协议。

读写 SPI FLASH API

文件 ARC_SPI_FLASH.c

ARC_SPI_Flash.c

```
377
378 /**
379   * @brief  Reads a block of data from the FLASH.
380   * @param  pBuffer: pointer to the buffer that receives the data read from the FLASH.
381   * @param  ReadAddr: FLASH's internal address to read from.
382   * @param  NumByteToRead: number of bytes to read from the FLASH.
383   * @retval None
384   */
385 void ARC_FLASH_ReadBuffer(uint8_t* pBuffer, uint32_t ReadAddr, uin
386 {
387     /*!< Select the FLASH: Chip Select low */
388     ARC_FLASH_CS_LOW();
389
390     /*!< Send "Read from Memory " instruction */
391     ARC_SPI_SendByte(SPI1, spi_flash_list[spi_flash_index].flash_cmd_read);
392
393     /*!< Send ReadAddr high nibble address byte to read from */
394     ARC_SPI_SendByte(SPI1, (ReadAddr & 0xFF0000) >> 16);
395     /*!< Send ReadAddr medium nibble address byte to read from */
396     ARC_SPI_SendByte(SPI1, (ReadAddr& 0xFF00) >> 8);
397     /*!< Send ReadAddr low nibble address byte to read from */
398     ARC_SPI_SendByte(SPI1, ReadAddr & 0xFF);
399
400     while (NumByteToRead--) /*!< while there is data to be read */
401     {
402         /*!< Read a byte from the FLASH */
403         *pBuffer = ARC_SPI_SendByte(SPI1, FLASH_DUMMY_BYTE);
404         /*!< Point to the next location where the byte read will be saved */
405         pBuffer++;
406     }
407
408     /*!< Deselect the FLASH: Chip Select high */
409     ARC_FLASH_CS_HIGH();
410 }
411
```

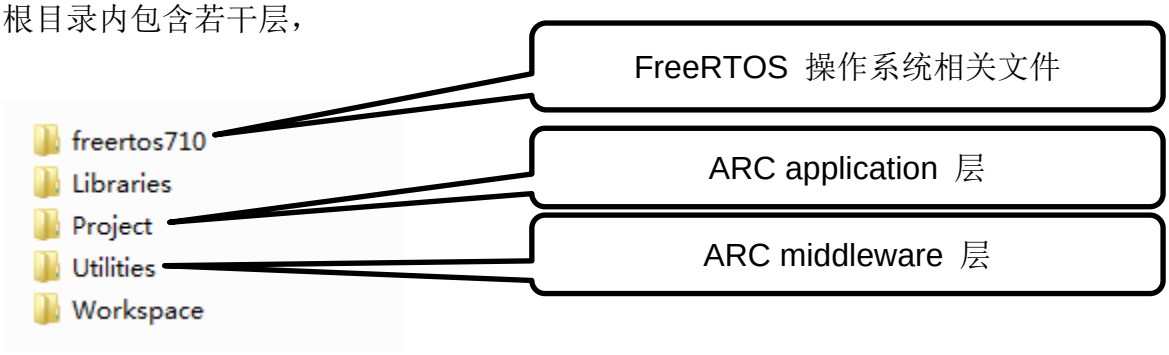
读 SPI flash API

该代码模块化很好，要支持其它 flash, 参考 flash 的规格书，填入下面的结构即可。

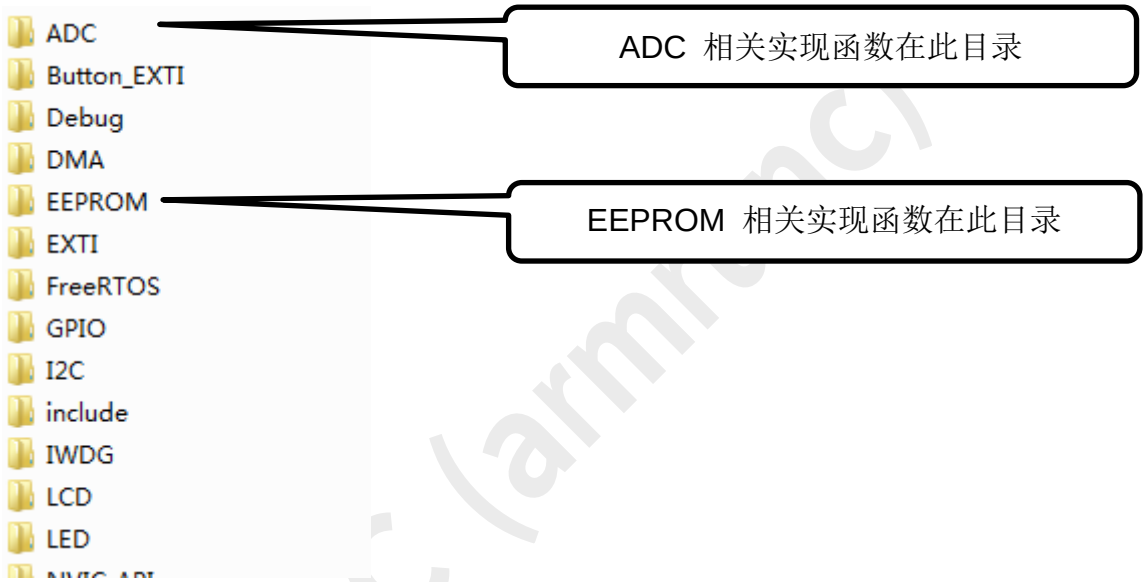
```
/* SPI flash list, add the flash you want to support here */
SPI_FLASH_CMD spi_flash_list[]=
{
    {
        0x9f,  /*!< Read flash id instruction */
        FLASH_W25Q16,
        "Winbond W25Q16",
        0x02,  /*!< Page program instruction */
        0x06,  /*!< Write enable instruction */
        0x03,  /*!< Read from Memory instruction */
        0x05,  /*!< Read Status Register instruction  */
        0x20,  /*!< Sector Erase instruction */
        0xD8,  /*!< Block Erase instruction */
        0xC7,  /*!< Chip Erase instruction */
        0x100, /*!< Chip page size */
    },
    {
        0x9f,  /*!< Read flash id instruction */
        FLASH_W25X40,
        "Winbond W25X40",
        0x02,  /*!< Page program instruction */
        0x06,  /*!< Write enable instruction */
        0x03,  /*!< Read from Memory instruction */
        0x05,  /*!< Read Status Register instruction  */
        0x20,  /*!< Sector Erase instruction */
        0xD8,  /*!< Block Erase instruction */
        0xC7,  /*!< Chip Erase instruction */
        0x100, /*!< Chip page size */
    }
};
```

文件目录安排也非常清晰明了，

根目录内包含若干层，



应用层的目录结构



应用层相关结构如下：