# STM32 启动代码概述

一般嵌入式开发流程就是先建立一个工程，再编写源文件，然后进行编译，把所有的*.s文件和*.c文件编译成一个*.o文件，再对目标文件进行链接和定位，编译成功后会生成一个*.hex文件和调试文件，接下来要进行调试，如果成功的话，就可以将它固化到flash里面去。

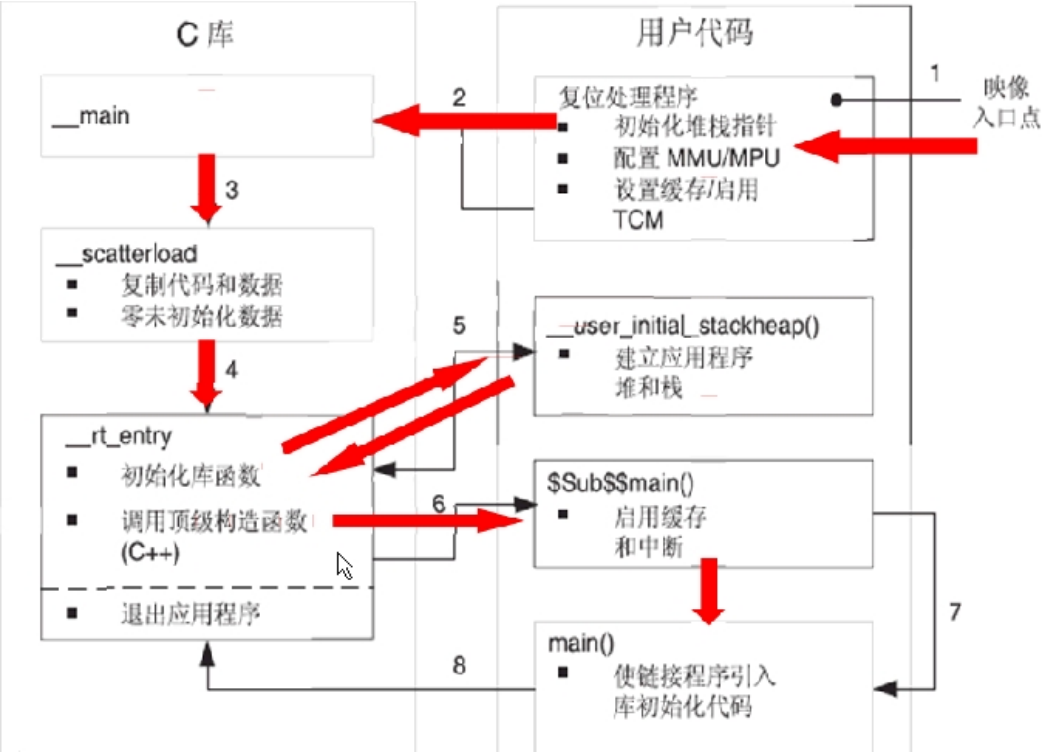
启动代码是用来初始化电路以及用来为高级语言写的软件作好运行前准备的一小段汇编语言，是任何处理器上电复位时的程序运行入口点。

比如，刚上电的过程中，PC机会对系统的一个运行频率进行锁定在一个固定的值，这个设计频率的过程就是在汇编源代码中进行的，也就是在启动代码中进行的。与此同时，设置完后，程序开始运行，注意，程序是在内存中运行的。这个时候，就需要把一些源文件从flash里面copy到内存中，又要对它们进行初始化读写，这又有频率的设置。这些都是初始化。

初始化完成后，我们又要设置一些堆栈，要跳到C语言的main函数里面运行。这就需要堆栈。对普通的ARM CPU有这样一个要求：在绝对地址为零的地方要放置一个异常向量表，但并不是所有的ARM CPU都留有这个一个空间，这就需要用到映射的功能。我们可以将其它地方的一些空间映射到绝对地址里面。当发生异常时，ARM核来读取异常中断表的时候，它会使用映射之后的那个表，这个就可以接着往下执行，否则在绝对地址零的地方找不到任何信息，程序就会死掉。这些运行的环境全部建立好后，程序就会跳转到我们的main函数里面。

总之，启动代码，就是对最小系统的初始化。包括晶振，CPU 频率等。

启动代码的最小系统是：异常向量表的初始化－存储区分配－初始化堆栈－高级语言入口函数调用－ main()函数。程序的启动过程：




以下面这个例子为例，编译完后，DEBUG后，我们可以看到，光标指向绝对地址为零的地方，这里存放的就是一个异常向量表。

```
0x00000000  E59FF018  LDR     PC,[PC,#0x0018]
0x00000004  E59FF018  LDR     PC,[PC,#0x0018]
0x00000008  E59FF018  LDR     PC,[PC,#0x0018]
0x0000000C  E59FF018  LDR     PC,[PC,#0x0018]
0x00000010  E59FF018  LDR     PC,[PC,#0x0018]
0x00000014  E1A00000  NOP
0x00000018  E51FFFF0  LDR     PC,[PC,#-0x0FF0]
0x0000001C  E59FF018  LDR     PC,[PC,#0x0018]
```

它对应在 startup.s 里的源文件如下：

```
158  Vectors        LDR    PC, Reset_Addr
159                 LDR    PC, Undef_Addr
160                 LDR    PC, SWI_Addr
161                 LDR    PC, PAbt_Addr
162                 LDR    PC, DAbt_Addr
163                 NOP                          ; Reserved Vector
164  ;              LDR    PC, IRQ_Addr
165                 LDR    PC, [PC, #-0x0FF0]    ; Vector from VicVectAddr
166                 LDR    PC, FIQ_Addr
```

单步运行后，马上跳转到初始化 CPU 的频率。即初始化锁相环，将其锁在一个固定的频率。具体代码如下：

```
; Setup PLL
        IF    PLL_SETUP <> 0
        LDR   R0, =PLL_BASE
        MOV   R1, #0xAA
        MOV   R2, #0x55

; Configure and Enable PLL
        MOV   R3, #PLLCFG_Val
        STR   R3, [R0, #PLLCFG_OFS]
        MOV   R3, #PLLCON_PLLE
        STR   R3, [R0, #PLLCON_OFS]
        STR   R1, [R0, #PLLFEED_OFS]
        STR   R2, [R0, #PLLFEED_OFS]

; Wait until PLL Locked
PLL_Loop    LDR   R3, [R0, #PLLSTAT_OFS]
        ANDS  R3, R3, #PLLSTAT_PLOCK
        BEQ   PLL_Loop
```

; Switch to PLL Clock

```
MOV    R3, #(PLLCON_PLLE:OR:PLLCON_PLLC)
STR    R3, [R0, #PLLCON_OFS]
STR    R1, [R0, #PLLFEED_OFS]
STR    R2, [R0, #PLLFEED_OFS]
ENDIF  ; PLL_SETUP
```

然后再初始化每一种模式的堆栈，再进行单步运行的时候，下面我们可以看到，它自动跳转到main（）函数：

; Enter the C code

```
IMPORT __main
LDR    R0, =__main
BX     R0

 IF     :DEF:__MICROLIB

EXPORT __heap_base
EXPORT __heap_limit

 ELSE
```

这个时候，程序会运行各种scatterload函数，将我们的堆栈、全局变量等内容拷贝到内存中去。拷贝完后，就正式跳转到我们的main()函数中来执行了。

```
0x00000294  E8BD8070  LDMIA     R13!,{R4-R6,PC}
    51: int main (void) {
    52:   unsigned int n;
    53:
0x00000298  E92D4010  STMDB     R13!,{R4,R14}
    54:   IODIR1 = 0x00FF0000;                    /* P1.16..23
0x0000029C  E3A008FF  MOV       R0,#0x00FF0000
0x000002A0  E59F105C  LDR       R1,[PC,#0x005C]
0x000002A4  E5810018  STR       R0,[R1,#0x0018]
    55:   ADCR   = 0x002E0401;                    /* Setup A/D:
```

这就是启动代码执行的全过程，呵呵，平时我们看到以为只是执行 main()函数就行了，是不是没有想到在执行 main() 函数后还有这么多学问呢？

# STM32 启动代码分析[转]

启动代码文件名是 STM32F10X.S，它的作用先总结下，然后再分析。启动代码作用一般是：1）堆和栈的初始化；2）向量表定义；3）地址重映射及中断向量表的转移；4）设置系统时钟频率；5）中断寄存器的初始化；6）进入 C 应用程序。

（1）按启动代码的次序，先看堆和栈的初始化：

```
Stack_Size          EQU         0x00000200              ；定义 Stack_Size 为 0x00000200

                            AREA        STACK, NOINIT, READWRITE, ALIGN=3     ；定义栈，可
初始为 0，8 字节对齐
Stack_Mem           SPACE     Stack_Size              ；分配 0x200 个连续字节，并初始化为 0
__initial_sp     ；汇编代码地址标号



Heap_Size           EQU         0x00000000

                            AREA        HEAP, NOINIT, READWRITE, ALIGN=3
__heap_base
Heap_Mem            SPACE     Heap_Size
__heap_limit


                            PRESERVE8  ；指定当前文件堆栈 8 字节对齐
                            THUMB          ；告诉汇编器下面是 32 为的 Thumb 指令，如果需要汇
编器将插入位以保证对齐
```

（2）中断向量表定义

```
AREA        RESET, DATA, READONLY ;定义复位向量段，只读
          EXPORT    __Vectors      ；定义一个可以在其他文件中使用的全局标号。此处表示中断地址

__Vectors       DCD      __initial_sp            ; 给__initial_sp 分配 4 字节 32 位的地址 0x0
            DCD      Reset_Handler           ; 给标号 Reset Handler 分配地址为 0x00000004
            DCD      NMI_Handler             ; 给标号 NMI Handler 分配地址 0x00000008
            DCD      HardFault_Handler       ; Hard Fault Handler
            DCD      MemManage_Handler       ; MPU Fault Handler
            DCD      BusFault_Handler        ; Bus Fault Handler
            DCD      UsageFault_Handler      ; Usage Fault Handler
            DCD      0                       ; 这种形式就是保留地址，不给任何标号分配
            DCD      0                       ; Reserved
            DCD      0                       ; Reserved
            DCD      0                       ; Reserved
          DCD     SVC_Handler             ; SVCall Handler
```

```
        DCD     DebugMon_Handler        ; Debug Monitor Handler
        DCD     0               ; Reserved
        DCD     PendSV_Handler          ; PendSV Handler
        DCD     SysTick_Handler         ; SysTick Handler

                                ; External Interrupts
                                DCD         WWDG_IRQHandler                     ; Window Watc
hdog
                                DCD         PVD_IRQHandler                      ; PVD throug
h EXTI Line detect
                                DCD         TAMPER_IRQHandler                 ; Tamper
                                DCD         RTC_IRQHandler                      ; RTC
                                DCD         FLASH_IRQHandler                  ; Flash
                                DCD         RCC_IRQHandler                      ; RCC
                                DCD         EXTI0_IRQHandler                  ; EXTI Line 0
                                DCD         EXTI1_IRQHandler                  ; EXTI Line 1
                                DCD         EXTI2_IRQHandler                  ; EXTI Line 2
                                DCD         EXTI3_IRQHandler                  ; EXTI Line 3
                                DCD         EXTI4_IRQHandler                  ; EXTI Line 4
                                DCD         DMAChannel1_IRQHandler       ; DMA Channel 1
                                DCD         DMAChannel2_IRQHandler       ; DMA Channel 2
                                DCD         DMAChannel3_IRQHandler       ; DMA Channel 3
                                DCD         DMAChannel4_IRQHandler       ; DMA Channel 4
                                DCD         DMAChannel5_IRQHandler       ; DMA Channel 5
                                DCD         DMAChannel6_IRQHandler       ; DMA Channel 6
                                DCD         DMAChannel7_IRQHandler       ; DMA Channel 7
                                DCD         ADC_IRQHandler                      ; ADC
                                DCD         USB_HP_CAN_TX_IRQHandler   ; USB High Priority or
 CAN TX
                                DCD         USB_LP_CAN_RX0_IRQHandler ; USB Low   Priority or
 CAN RX0
                                DCD         CAN_RX1_IRQHandler               ; CAN RX1
                                DCD         CAN_SCE_IRQHandler               ; CAN SCE
                                DCD         EXTI9_5_IRQHandler               ; EXTI Line 9..5
                                DCD         TIM1_BRK_IRQHandler             ; TIM1 Break
                                DCD         TIM1_UP_IRQHandler               ; TIM1 Update
                                DCD         TIM1_TRG_COM_IRQHandler     ; TIM1 Trigger and Co
mmutation
                                DCD         TIM1_CC_IRQHandler               ; TIM1 Capture C
ompare
                                DCD         TIM2_IRQHandler                     ; TIM2
                                DCD         TIM3_IRQHandler                     ; TIM3
                                DCD         TIM4_IRQHandler                     ; TIM4
                                DCD         I2C1_EV_IRQHandler               ; I2C1 Event
```

```
                DCD         I2C1_ER_IRQHandler              ; I2C1 Error
                DCD         I2C2_EV_IRQHandler              ; I2C2 Event
                DCD         I2C2_ER_IRQHandler              ; I2C2 Error
                DCD         SPI1_IRQHandler                    ; SPI1
                DCD         SPI2_IRQHandler                    ; SPI2
                DCD         USART1_IRQHandler                ; USART1
                DCD         USART2_IRQHandler                ; USART2
                DCD         USART3_IRQHandler                ; USART3
                DCD         EXTI15_10_IRQHandler           ; EXTI Line 15..10
                DCD         RTCAlarm_IRQHandler             ; RTC Alarm throu
gh EXTI Line
                DCD         USBWakeUp_IRQHandler           ; USB Wakeup from
suspend
```

（3）中断向量表的转移

```
AREA        |.text|, CODE, READONLY ；代码段定义


; Reset Handler

Reset_Handler     PROC  ；标记一个函数的开始
                EXPORT   Reset_Handler                        [WEAK]；[WEAK]
```

选项表示当所有的源文件都没有 定义这样一个标号时，编译器也不给出错误信息，在多数情况下将该标号置为 0 ，若该标号为 B 或 BL 指令引用，则将 B 或 BL 指令置为 NOP 操作。EXPORT 提示编译器该标号可以为外部文件引用。

```
                IMPORT   __main ;通知编译器要使用的标号在其他文件
                LDR         R0, =__main；使用“=”表示 LDR 目前是伪指令不是标准
```

指令。这里是把__main 的地址给 R0。

```
                BX           R0；BX 是 ARM 指令集和 THUMB 指令集之间程序的跳转
                ENDP


; Dummy Exception Handlers (infinite loops which can be modifie
d)

NMI_Handler         m    ;"m"其实就是 PROC 表示汇编函数的开始
                EXPORT   NMI_Handler                          [WEAK]
                B            .
                ENDP
HardFault_Handler\          ;"\"是换行的意思
                PROC
                EXPORT   HardFault_Handler                [WEAK]
                B            . ;"."号到底是什么含义呢，目前还没查到资料。可能
```

是保留地址，供以后修改的吧

```
                                ENDP
MemManage_Handler\
                                PROC
                                EXPORT   MemManage_Handler                  [WEAK]
                                B              .
                                ENDP
BusFault_Handler\
                                PROC
                                EXPORT   BusFault_Handler                   [WEAK]
                                B              .
                                ENDP
UsageFault_Handler\
                                PROC
                                EXPORT   UsageFault_Handler                [WEAK]
                                B              .
                                ENDP
SVC_Handler          PROC
                                EXPORT   SVC_Handler                              [WEAK]
                                B              .
                                ENDP
DebugMon_Handler\
                                PROC
                                EXPORT   DebugMon_Handler                   [WEAK]
                                B              .
                                ENDP
PendSV_Handler    PROC
                                EXPORT   PendSV_Handler                       [WEAK]
                                B              .
                                ENDP
SysTick_Handler PROC
                                EXPORT   SysTick_Handler                    [WEAK]
                                B              .
                                ENDP

Default_Handler PROC

                                EXPORT   WWDG_IRQHandler                    [WEAK]
                                EXPORT   PVD_IRQHandler                       [WEAK]
                                EXPORT   TAMPER_IRQHandler                 [WEAK]
                                EXPORT   RTC_IRQHandler                       [WEAK]
                                EXPORT   FLASH_IRQHandler                   [WEAK]
                                EXPORT   RCC_IRQHandler                       [WEAK]
                                EXPORT   EXTI0_IRQHandler                   [WEAK]
                                EXPORT   EXTI1_IRQHandler                   [WEAK]
```

```
                EXPORT   EXTI2_IRQHandler                    [WEAK]
                EXPORT   EXTI3_IRQHandler                    [WEAK]
                EXPORT   EXTI4_IRQHandler                    [WEAK]
                EXPORT   DMAChannel1_IRQHandler       [WEAK]
                EXPORT   DMAChannel2_IRQHandler       [WEAK]
                EXPORT   DMAChannel3_IRQHandler       [WEAK]
                EXPORT   DMAChannel4_IRQHandler       [WEAK]
                EXPORT   DMAChannel5_IRQHandler       [WEAK]
                EXPORT   DMAChannel6_IRQHandler       [WEAK]
                EXPORT   DMAChannel7_IRQHandler       [WEAK]
                EXPORT   ADC_IRQHandler                       [WEAK]
                EXPORT   USB_HP_CAN_TX_IRQHandler   [WEAK]
                EXPORT   USB_LP_CAN_RX0_IRQHandler [WEAK]
                EXPORT   CAN_RX1_IRQHandler               [WEAK]
                EXPORT   CAN_SCE_IRQHandler               [WEAK]
                EXPORT   EXTI9_5_IRQHandler               [WEAK]
                EXPORT   TIM1_BRK_IRQHandler             [WEAK]
                EXPORT   TIM1_UP_IRQHandler               [WEAK]
                EXPORT   TIM1_TRG_COM_IRQHandler     [WEAK]
                EXPORT   TIM1_CC_IRQHandler               [WEAK]
                EXPORT   TIM2_IRQHandler                     [WEAK]
                EXPORT   TIM3_IRQHandler                     [WEAK]
                EXPORT   TIM4_IRQHandler                     [WEAK]
                EXPORT   I2C1_EV_IRQHandler               [WEAK]
                EXPORT   I2C1_ER_IRQHandler               [WEAK]
                EXPORT   I2C2_EV_IRQHandler               [WEAK]
                EXPORT   I2C2_ER_IRQHandler               [WEAK]
                EXPORT   SPI1_IRQHandler                     [WEAK]
                EXPORT   SPI2_IRQHandler                     [WEAK]
                EXPORT   USART1_IRQHandler                [WEAK]
                EXPORT   USART2_IRQHandler                [WEAK]
                EXPORT   USART3_IRQHandler                [WEAK]
                EXPORT   EXTI15_10_IRQHandler           [WEAK]
                EXPORT   RTCAlarm_IRQHandler             [WEAK]
                EXPORT   USBWakeUp_IRQHandler           [WEAK]

WWDG_IRQHandler
PVD_IRQHandler
TAMPER_IRQHandler
RTC_IRQHandler
FLASH_IRQHandler
RCC_IRQHandler
EXTI0_IRQHandler
EXTI1_IRQHandler
```

```
EXTI2_IRQHandler
EXTI3_IRQHandler
EXTI4_IRQHandler
DMAChannel1_IRQHandler
DMAChannel2_IRQHandler
DMAChannel3_IRQHandler
DMAChannel4_IRQHandler
DMAChannel5_IRQHandler
DMAChannel6_IRQHandler
DMAChannel7_IRQHandler
ADC_IRQHandler
USB_HP_CAN_TX_IRQHandler
USB_LP_CAN_RX0_IRQHandler
CAN_RX1_IRQHandler
CAN_SCE_IRQHandler
EXTI9_5_IRQHandler
TIM1_BRK_IRQHandler
TIM1_UP_IRQHandler
TIM1_TRG_COM_IRQHandler
TIM1_CC_IRQHandler
TIM2_IRQHandler
TIM3_IRQHandler
TIM4_IRQHandler
I2C1_EV_IRQHandler
I2C1_ER_IRQHandler
I2C2_EV_IRQHandler
I2C2_ER_IRQHandler
SPI1_IRQHandler
SPI2_IRQHandler
USART1_IRQHandler
USART2_IRQHandler
USART3_IRQHandler
EXTI15_10_IRQHandler
RTCAlarm_IRQHandler
USBWakeUp_IRQHandler

                                B               .

                                ENDP
                                ALIGN
```

（4）堆和栈的初始化

```
IF          :DEF:__MICROLIB              ；“DEF”的用法——:DEF:X 就是说 X 定义了则为真，否则为假

                            EXPORT   __initial_sp
                            EXPORT   __heap_base
                            EXPORT   __heap_limit

                            ELSE

                            IMPORT   __use_two_region_memory
                            EXPORT   __user_initial_stackheap
__user_initial_stackheap

                            LDR         R0, =   Heap_Mem
                            LDR         R1, =(Stack_Mem + Stack_Size)
                            LDR         R2, = (Heap_Mem +   Heap_Size)
                            LDR         R3, = Stack_Mem
                            BX           LR

                            ALIGN ；填充字节使地址对齐

                            ENDIF

END
```