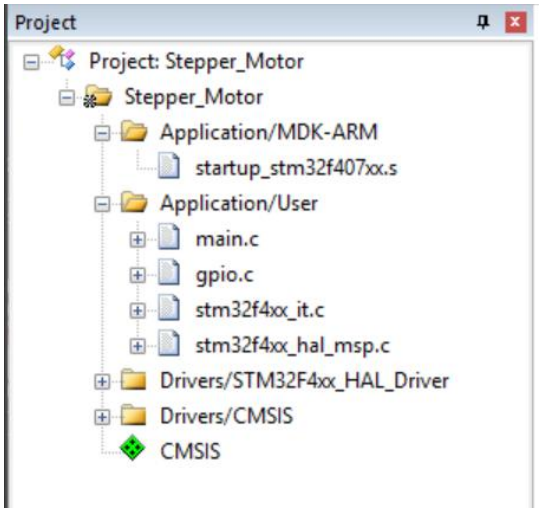
步进电机：

1. 工程文件分析

Project 工程树情况如下图：



其中，各个节点之间的关以及各自的作用如下：

Project: Stepper_Motor

Application/MDK-ARM　　# 启动文件（startup_stm32f407xx.s）

Application/User　　# 用户代码

　　├─ main.c　　# 主逻辑（八拍控制、延时）

　　├─ gpio.c　　# GPIO 初始化

　　├─ stm32f4xx_it.c　　# 中断服务函数（未修改，默认空）

　　└─ stm32f4xx_hal_msp.c　# HAL 库底层初始化（时钟、外设）

Drivers/STM32F4xx_HAL_Driver　# ST 官方 HAL 库（GPIO、延时等驱

动）

Drivers/CMSIS　　# 芯片支持包（启动文件、寄存器定义）

2. startup 启动代码.s 文件剖析：

```
33  Stack_Size      EQU     0x400
34
35                  AREA    STACK, NOINIT, READWRITE, ALIGN=3
36  Stack_Mem       SPACE   Stack_Size
37  __initial_sp
38
39
40  ; <h> Heap Configuration
41  ;   <o>  Heap Size (in Bytes) <0x0-0xFFFFFFFF:8>
42  ; </h>
43
44  Heap_Size       EQU     0x200
45
46                  AREA    HEAP, NOINIT, READWRITE, ALIGN=3
47  __heap_base
48  Heap_Mem        SPACE   Heap_Size
49  __heap_limit
50
51                  PRESERVE8
52                  THUMB
53
```

这段代码是堆栈与堆初始化，功能：定义主堆栈（向下生长）和堆的内存区域，初始化栈顶指针（__initial_sp），供 Cortex-M4 内核启动时使用。

```
55  ; Vector Table Mapped to Address 0 at Reset
56                  AREA    RESET, DATA, READONLY
57                  EXPORT  __Vectors
58                  EXPORT  __Vectors_End
59                  EXPORT  __Vectors_Size
60
61  __Vectors       DCD     __initial_sp              ; Top of Stack
62                  DCD     Reset_Handler             ; Reset Handler
63                  DCD     NMI_Handler               ; NMI Handler
64                  DCD     HardFault_Handler         ; Hard Fault Handler
65                  DCD     MemManage_Handler         ; MPU Fault Handler
66                  DCD     BusFault_Handler          ; Bus Fault Handler
67                  DCD     UsageFault_Handler        ; Usage Fault Handler
68                  DCD     0                         ; Reserved
69                  DCD     0                         ; Reserved
70                  DCD     0                         ; Reserved
71                  DCD     0                         ; Reserved
72                  DCD     SVC_Handler               ; SVCall Handler
73                  DCD     DebugMon_Handler          ; Debug Monitor Handler
74                  DCD     0                         ; Reserved
75                  DCD     PendSV_Handler            ; PendSV Handler
```

这段代码是原来定义向量表的，功能：①定义中断向量表，位于 Flash 起始地址（0x08000000），每个向量占 4 字节（存储中断服务程序 ISR 的地址）。②前 16 项为内核异常（如复位、NMI），后续为外设中断（如 GPIO、USART）。③复位后，CPU 从 0x08000004 读取 Reset_Handler 地址，开始执行启动流程。

```
169  ; Reset handler
170  Reset_Handler    PROC
171                   EXPORT  Reset_Handler             [WEAK]
172          IMPORT  SystemInit
173          IMPORT  __main
174
175                   LDR     R0, =SystemInit
176                   BLX     R0
177                   LDR     R0, =__main
178                   BX      R0
179                   ENDP
180
```

这段是复位处理函数，功能：①初始化系统时钟（通过 SystemInit，位于 system_stm32f4xx.c）。②跳转至 C 库__main，完成全局变量初始化（.data/.bss 段），最终进入用户 main()。

```
181  ; Dummy Exception Handlers (infinite loops which can be modified)
182
183  NMI_Handler     PROC
184                  EXPORT  NMI_Handler               [WEAK]
185                  B       .
186                  ENDP
187  HardFault_Handler\
188                  PROC
189                  EXPORT  HardFault_Handler         [WEAK]
190                  B       .
191                  ENDP
192  MemManage_Handler\
193                  PROC
194                  EXPORT  MemManage_Handler         [WEAK]
195                  B       .
196                  ENDP
197  BusFault_Handler\
198                  PROC
199                  EXPORT  BusFault_Handler          [WEAK]
200                  B       .
201                  ENDP
```

这段代码是默认中断处理函数，功能：

① 所有中断处理函数均为弱定义（[WEAK]），默认实现为无限循

环。

②用户需在.c 文件中定义同名函数（如 void USART1_IRQHandler(void)）以覆盖，否则触发默认空处理。

作用：避免因未实现中断函数导致的链接错误，确保工程编译通过。

```
; User Stack and Heap initialization
;*******************************************************************************
                IF      :DEF:__MICROLIB

                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
```

这段代码是内存初始化

功能：①配置堆栈和堆的内存区域，供 C 库初始化使用。②__MICROLIB 为 MDK 默认微库，无需额外初始化；标准库需通过 __user_initial_stackheap 指定内存边界。

其他关键段的功能如下

PRESERVE8：保证后续指令按 8 字节对齐，避免 Thumb 指令冲突。

THUMB：声明使用 Thumb 指令集（STM32F4 仅支持 Thumb-2）。

ALIGN：确保代码段对齐到 4 字节，优化指令执行效率。

对核心代码逐句加注释：

Stack/Heap Configuration：

```
;*******************************************************************************
;* <<< Use Configuration Wizard in Context Menu >>>
;
; Amount of memory (in bytes) allocated for Stack
; Tailor this value to your application needs
; <h> Stack Configuration
;   <o> Stack Size (in Bytes) <0x0-0xFFFFFFFF:8>
; </h>
Stack_Size      EQU     0x400   ; 栈大小：1KB（用户可根据需求调整）
                AREA    STACK, NOINIT, READWRITE, ALIGN=3
Stack_Mem       SPACE   Stack_Size
__initial_sp    ; 初始栈顶地址（向量表第1项）


; <h> Heap Configuration
;   <o>  Heap Size (in Bytes) <0x0-0xFFFFFFFF:8>
; </h>
Heap_Size       EQU     0x200   ; 堆大小：512B（用于动态内存分配）
                AREA    HEAP, NOINIT, READWRITE, ALIGN=3
__heap_base     ; 堆起始地址
Heap_Mem        SPACE   Heap_Size
__heap_limit    ; 堆结束地址



                PRESERVE8
                THUMB
```

Vector Table Mapped to Address 0 at Reset：

```
; Vector Table Mapped to Address 0 at Reset

                AREA    RESET, DATA, READONLY
                EXPORT  Vectors,  Vectors_End,  Vectors_Size
__Vectors       DCD     __initial_sp              ; 栈顶（0x08000000）
                DCD     Reset_Handler             ; 复位向量（0x08000004）
                DCD     NMI_Handler               ; 不可屏蔽中断
                DCD     HardFault_Handler         ; 硬错误中断
```

```
__Vectors_End
__Vectors_Size  EQU     __Vectors_End - __Vectors   ; 向量表大小（368字节，92×4）
```

Reset handler：

```
; Reset handler
Reset_Handler   PROC
                EXPORT  Reset_Handler [WEAK]   ；弱符号，允许用户覆盖
                IMPORT  SystemInit,   main     ；导入系统时钟初始化和C库入口
                LDR     R0, =SystemInit        ；加载SystemInit地址
                BLX     R0                     ；跳转到SystemInit（初始化时钟）
                LDR     R0, =   main           ；加载C库入口  main
                BX      R0                     ；跳转至  main（最终调用用户main()）
                ENDP
```

Dummy Exception Handlers：

```
; Dummy Exception Handlers (infinite loops which can be modified)
NMI_Handler     PROC    ；不可屏蔽中断（如电压异常）
                EXPORT  NMI_Handler [WEAK]
                B       .                  ；无限循环（用户需自定义）
                ENDP
HardFault_Handler\    ；硬错误（如总线错误、用法错误）
                PROC
                EXPORT  HardFault_Handler [WEAK]
                B       .
                ENDP  ；（其他中断均类似，如USART1_IRQHandler、TIM2_IRQHandler）
```

User Stack and Heap initialization：

```
;*******************************************************************************
; User Stack and Heap initialization
;*******************************************************************************
                IF      :DEF:  MICROLIB    ；使用微库（MDK默认）
                EXPORT    initial_sp,   heap_base,   heap_limit
                ELSE                       ；使用标准库
                IMPORT    use_two_region_memory
                EXPORT    user_initial_stackheap
  user_initial_stackheap
                LDR     R0, =Heap_Mem      ；堆起始地址
                LDR     R1, =(Stack_Mem + Stack_Size) ；栈结束地址
                LDR     R2, =(Heap_Mem + Heap_Size)  ；堆结束地址
                LDR     R3, =Stack_Mem     ；栈起始地址
                BX      LR                 ；返回，触发C库内存初始化
                ALIGN
                ENDIF
```

3. 步进电机实验原理解析（结合 main.c 和 gpio.c）

①GPIO 初始化（gpio.c）：

引脚分配（对应四相步进电机）

电机相线	STM32 引脚	功能
A 相	GPIOD_PIN_12	控制绕组 A 的通断
B 相	GPIOH_PIN_13	控制绕组 B 的通断
C 相	GPIOE_PIN_4	控制绕组 C 的通断
D 相	GPIOE_PIN_6	控制绕组 D 的通断

初始化代码：

__HAL_RCC_GPIOE_CLK_ENABLE(); //使能 GPIOE 时钟（C、D 相）

__HAL_RCC_GPIOH_CLK_ENABLE(); //使能 GPIOH 时钟（B 相）

__HAL_RCC_GPIOD_CLK_ENABLE(); //使能 GPIOD 时钟（A 相）

GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP; //推挽输出（灌电流驱动 ULN2003）

② 八拍励磁控制（main.c）

#define DE_A　　A 相高，其余低　// 1000（假设 A-B-C-D 相序）

#define DE_AB　A+B 相高　　// 1100

#define DE_B　　B 相高　　// 0100

#define DE_BC　B+C 相高　　// 0110

#define DE_C　　C 相高　　// 0010

#define DE_CD　C+D 相高　　// 0011

#define DE_D　　D 相高　　// 0001

#define DE_DA　D+A 相高　　// 1001

原理：每拍切换两相（半步模式），步距角为整步的一半（如 1.8°→0.9°），转动更平滑。

转向控制：

顺时针：A→AB→B→BC→C→CD→D→DA（main.c 的第一个 while(1)）。

逆时针：DA→D→CD→C→BC→B→AB→A（第二个 while(1)，实际代码被注释，需取消注释切换方向）。

③ 转速控制：

通过 HAL_Delay(1)控制每拍间隔（如 1ms），延时越短转速越快。

④ 硬件电路驱动：

ULN2003 驱动：STM32 GPIO 输出控制 ULN2003 的 IN1-IN4，其内部达林顿管放大电流（≥500mA），驱动步进电机绕组。

电气隔离：ULN2003 自带续流二极管，保护 STM32 免受电机反电动势冲击。

总结：这个步进电机实验的原理是通过 GPIO 模拟八拍励磁，利用 HAL 库延时控制转速，结合 ULN2003 驱动实现步进电机的正反转