第6章 MCU应用程序设计

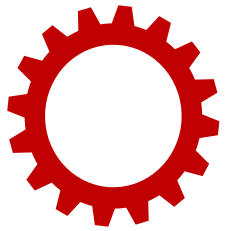
汇编程序设计？还是算了

C程序设计？不用重复了吧

编程？

也许---是表达思想
把你的想法表达给机器，
让机器按你的意愿工作，
达到你希望达到的目的

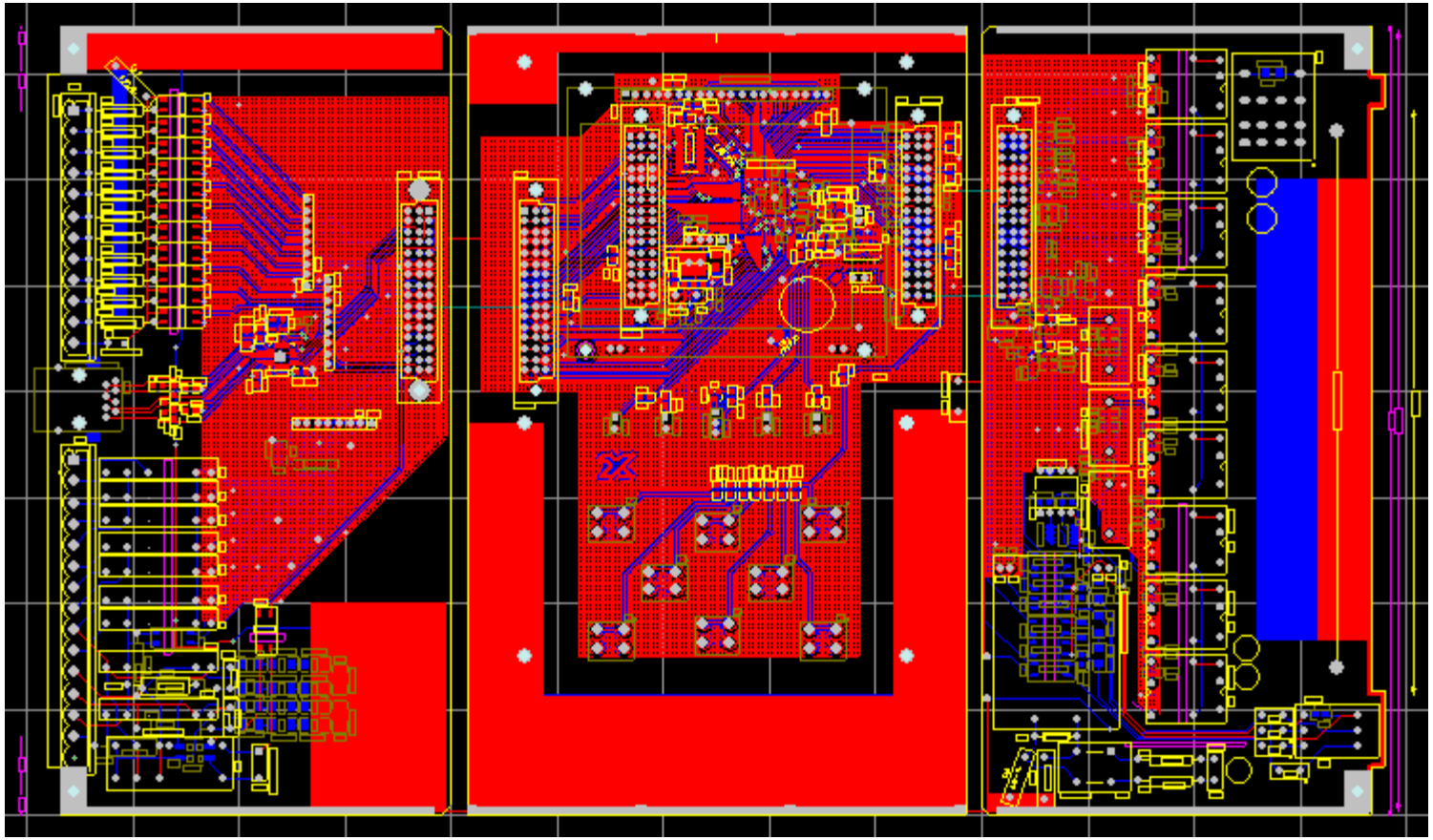
不编程可以吗？
可以，动动手指头，点点屏幕，
按按键盘、鼠标。
机器也在为我们服务



从一个实例观察

要做什么？
怎么做？
……

没有软件，这一切都
是没有意义的材料



1 一种开发工具

Features

- Intuitive STM32 microcontroller and microprocessor selection
- Rich easy-to-use graphical user interface allowing the configuration of:
  - Pinout with automatic conflict resolution
  - Peripherals and middleware functional modes with dynamic validation of parameter constraints for Arm® Cortex®-M core
  - Clock tree with dynamic validation of the configuration
  - Power sequence with estimated consumption results
- Generation of initialization C code project, compliant with IAR Embedded Workbench®, MDK-ARM and STM32CubeIDE (GCC compilers) for Arm® Cortex®-M core
- Generation of a partial Linux® Device Tree for Arm® Cortex®-A core (STM32 microprocessors)
- Development of enhanced STM32Cube Expansion Packages thanks to STM32PackCreator
- Integration of STM32Cube Expansion packages into the project
- Availability as standalone software running on Windows®, Linux® and macOS® (macOS® is a trademark of Apple Inc. registered in the U.S. and other countries.) operating systems and 64-bit Java Runtime environment

Categories | A->Z

Analog
- ADC1
- ADC2
- ADC3
- DAC

Timers
- RTC
- TIM1
- TIM2
- TIM3
- TIM4
- TIM5
- TIM6
- TIM7
- TIM8

Connectivity
- CAN
- FSMC
- I2C1
- I2C2
- SDIO
- SPI1
- SPI2
- SPI3
- UART4
- UART5
- USART1
- USART2
- USART3
- USB

ADC1 Mode and Configuration

Mode

- IN0
- IN1
- IN2
- IN3
- IN4
- IN5
- IN6
- IN7
- IN8
- IN9
- IN10

Configuration

Reset Configuration

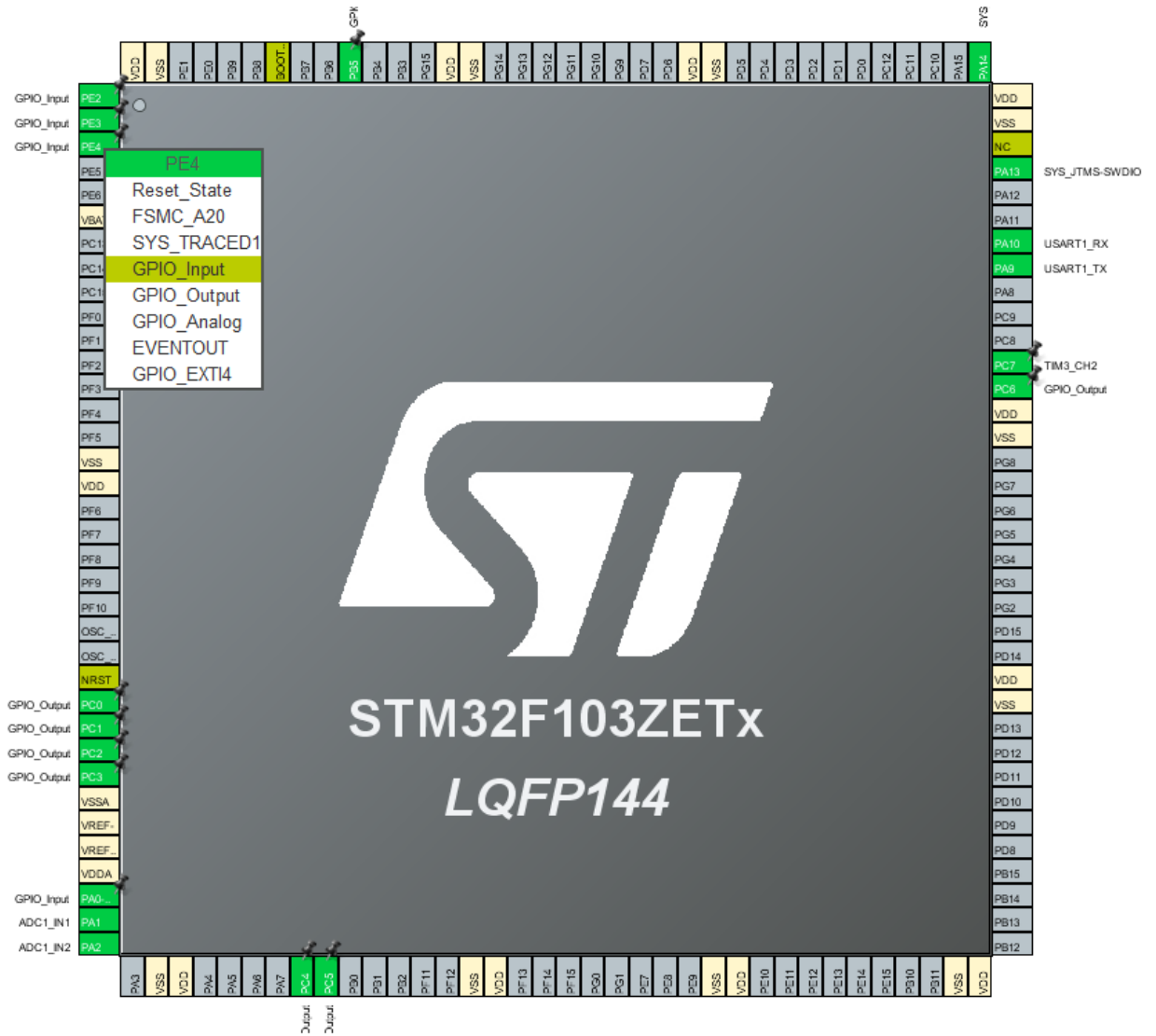
NVIC Settings | DMA Settings | GPIO Settings | Parameter Settings | User Constants

Configure the below parameters :

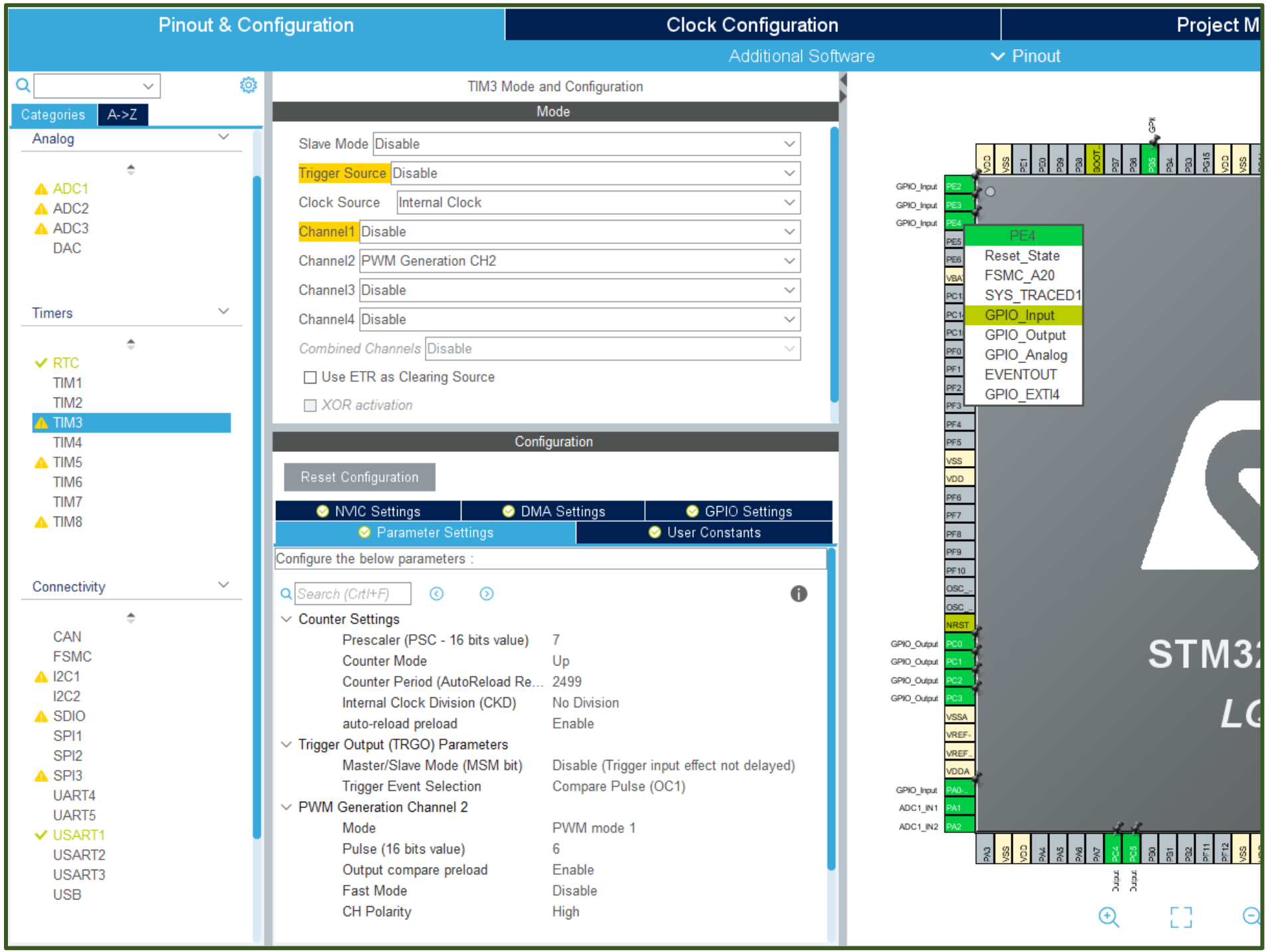
Search (Crtl+F)

ADCs_Common_Settings	
Mode	Independent mode
ADC_Settings	
Data Alignment	Right alignment
Scan Conversion Mode	Enabled
Continuous Conversion Mode	Disabled
Discontinuous Conversion Mode	Disabled
ADC_Regular_ConversionMode	
Enable Regular Conversions	Enable
Number Of Conversion	2
External Trigger Conversion Source	Timer 3 Trigger Out event
Rank	1
Rank	2
ADC_Injected_ConversionMode	

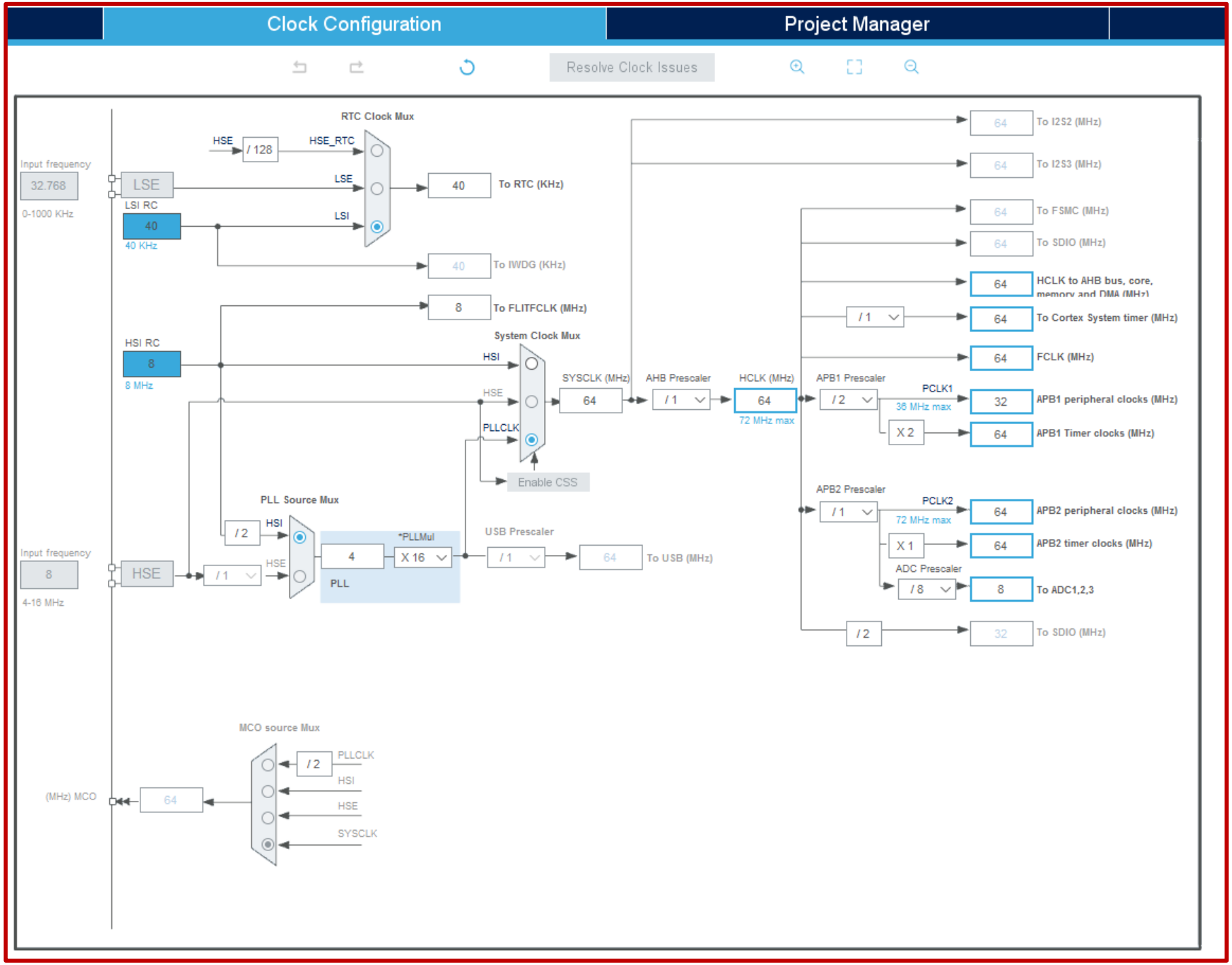
Pinout view | System view



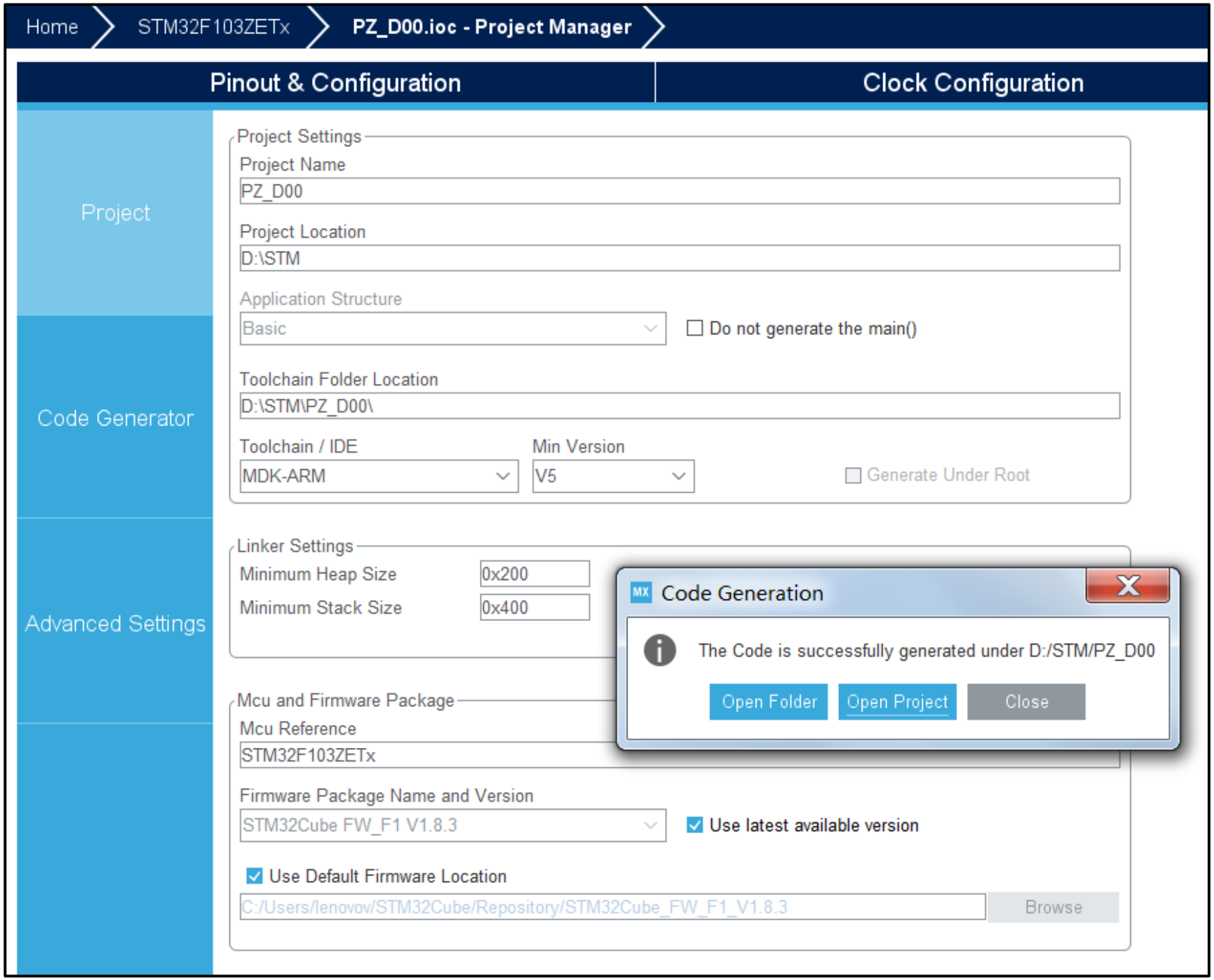
一些内容


Pinout & Configuration
Clock Configuration
Project M
Additional Software
Pinout
Categories
A->Z
Analog
ADC1
ADC2
ADC3
DAC
Timers
RTC
TIM1
TIM2
TIM3
TIM4
TIM5
TIM6
TIM7
TIM8
Connectivity
CAN
FSMC
I2C1
I2C2
SDIO
SPI1
SPI2
SPI3
UART4
UART5
USART1
USART2
USART3
USB
TIM3 Mode and Configuration
Mode
Slave Mode Disable
Trigger Source Disable
Clock Source Internal Clock
Channel1 Disable
Channel2 PWM Generation CH2
Channel3 Disable
Channel4 Disable
Combined Channels Disable
Use ETR as Clearing Source
XOR activation
Configuration
Reset Configuration
NVIC Settings
DMA Settings
GPIO Settings
Parameter Settings
User Constants
Configure the below parameters :
Search (Crtl+F)
Counter Settings
Prescaler (PSC - 16 bits value) 7
Counter Mode Up
Counter Period (AutoReload Re... 2499
Internal Clock Division (CKD) No Division
auto-reload preload Enable
Trigger Output (TRGO) Parameters
Master/Slave Mode (MSM bit) Disable (Trigger input effect not delayed)
Trigger Event Selection Compare Pulse (OC1)
PWM Generation Channel 2
Mode PWM mode 1
Pulse (16 bits value) 6
Output compare preload Enable
Fast Mode Disable
CH Polarity High
PE4
Reset_State
FSMC_A20
SYS_TRACED1
GPIO_Input
GPIO_Output
GPIO_Analog
EVENTOUT
GPIO_EXTI4

时钟相关


Clock Configuration
Project Manager
Resolve Clock Issues
RTC Clock Mux
HSE
/ 128
HSE_RTC
LSE
LSI
Input frequency
32.768
0-1000 KHz
LSE
LSI RC
40
40 KHz
40
To RTC (KHz)
40
To IWDG (KHz)
8
To FLITFCLK (MHz)
System Clock Mux
HSI RC
8
8 MHz
HSI
HSE
PLLCLK
Enable CSS
SYSCLK (MHz)
64
AHB Prescaler
/ 1
HCLK (MHz)
64
72 MHz max
APB1 Prescaler
/ 2
PCLK1
36 MHz max
X 2
APB2 Prescaler
/ 1
PCLK2
72 MHz max
X 1
ADC Prescaler
/ 8
PLL Source Mux
/ 2
HSI
HSE
/ 1
Input frequency
8
4-16 MHz
HSE
4
*PLLMul
X 16
PLL
USB Prescaler
/ 1
64
To USB (MHz)
64
To I2S2 (MHz)
64
To I2S3 (MHz)
64
To FSMC (MHz)
64
To SDIO (MHz)
64
HCLK to AHB bus, core, memory and DMA (MHz)
/ 1
64
To Cortex System timer (MHz)
64
FCLK (MHz)
32
APB1 peripheral clocks (MHz)
64
APB1 Timer clocks (MHz)
64
APB2 peripheral clocks (MHz)
64
APB2 timer clocks (MHz)
8
To ADC1,2,3
/ 2
32
To SDIO (MHz)
MCO source Mux
/ 2
PLLCLK
HSI
HSE
SYSCLK
(MHz) MCO
64

一些设定


Home
STM32F103ZETx
PZ_D00.ioc - Project Manager
Pinout & Configuration
Clock Configuration
Project
Code Generator
Advanced Settings
Project Settings
Project Name
PZ_D00
Project Location
D:\STM
Application Structure
Basic
Do not generate the main()
Toolchain Folder Location
D:\STM\PZ_D00\
Toolchain / IDE
MDK-ARM
Min Version
V5
Generate Under Root
Linker Settings
Minimum Heap Size
0x200
Minimum Stack Size
0x400
Mcu and Firmware Package
Mcu Reference
STM32F103ZETx
Firmware Package Name and Version
STM32Cube FW_F1 V1.8.3
Use latest available version
Use Default Firmware Location
C:/Users/lenovo/STM32Cube/Repository/STM32Cube_FW_F1_V1.8.3
Browse
Code Generation
The Code is successfully generated under D:/STM/PZ_D00
Open Folder
Open Project
Close

设计报告

示意

3. Pins Configuration

Pin Number LQFP144	Pin Name (function after reset)	Pin Type	Alternate Function(s)	Label
1	PE2 *	I/O	GPIO_Input	
2	PE3 *	I/O	GPIO_Input	
3	PE4 *	I/O	GPIO_Input	
6	VBAT	Power		
16	VSS	Power		
17	VDD	Power		
25	NRST	Reset		
26	PC0 *	I/O	GPIO_Output	
27	PC1 *	I/O	GPIO_Output	
28	PC2 *	I/O	GPIO_Output	
29	PC3 *	I/O	GPIO_Output	
30	VSSA	Power		
31	VREF-	Power		
32	VREF+	Power		
33	VDDA	Power		
34	PA0-WKUP *	I/O	GPIO_Input	
35	PA1	I/O	ADC1_IN1	
36	PA2	I/O	ADC1_IN2	
38	VSS	Power		
39	VDD	Power		
44	PC4 *	I/O	GPIO_Output	
45	PC5 *	I/O	GPIO_Output	
51	VSS	Power		
52	VDD	Power		
61	VSS	Power		
62	VDD	Power		
71	VSS	Power		
72	VDD	Power		
83	VSS	Power		
84	VDD	Power		
94	VSS	Power		
95	VDD	Power		

7. IPs and Middleware Configuration

7.1. ADC1

mode: IN1

mode: IN2

7.1.1. Parameter Settings:

ADCs_Common_Settings:

Mode	Independent mode

ADC_Settings:

Data Alignment	Right alignment
Scan Conversion Mode	Enabled
Continuous Conversion Mode	Disabled
Discontinuous Conversion Mode	Disabled

ADC_Regular_ConversionMode:

Enable Regular Conversions	Enable
Number Of Conversion	**2** *
External Trigger Conversion Source	**Timer 3 Trigger Out event** *
Rank	1
Channel	Channel 1
Sampling Time	**13.5 Cycles** *
Rank	**2** *
Channel	**Channel 2** *
Sampling Time	**13.5 Cycles** *

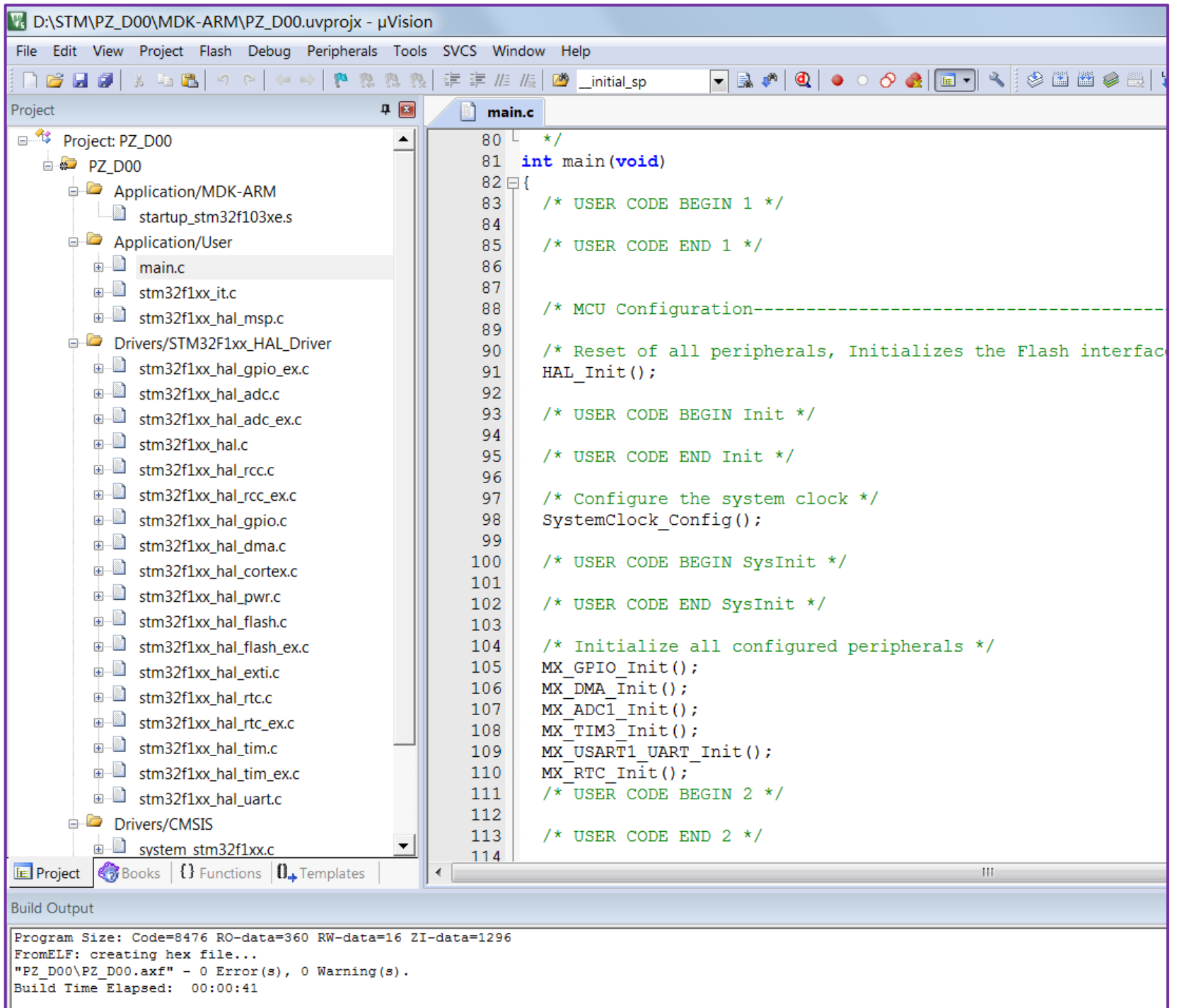
ADC_Injected_ConversionMode:

Enable Injected Conversions	Disable

WatchDog:

Enable Analog WatchDog Mode	false

生成的工程与代码
示意

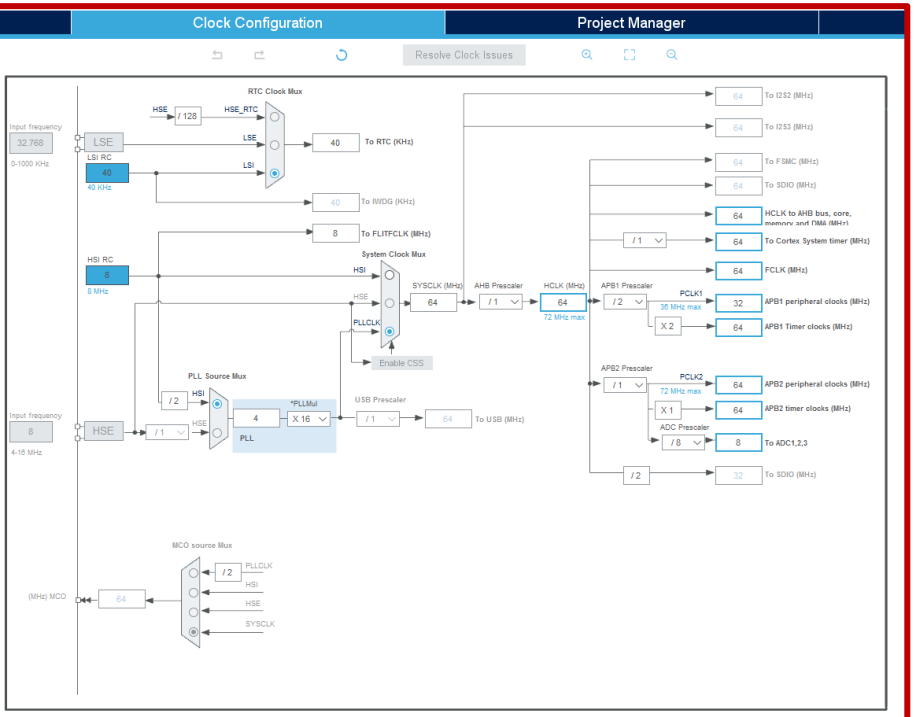

D:\STM\PZ_D00\MDK-ARM\PZ_D00.uvprojx - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
_initial_sp
Project
Project: PZ_D00
PZ_D00
Application/MDK-ARM
startup_stm32f103xe.s
Application/User
main.c
stm32f1xx_it.c
stm32f1xx_hal_msp.c
Drivers/STM32F1xx_HAL_Driver
stm32f1xx_hal_gpio_ex.c
stm32f1xx_hal_adc.c
stm32f1xx_hal_adc_ex.c
stm32f1xx_hal.c
stm32f1xx_hal_rcc.c
stm32f1xx_hal_rcc_ex.c
stm32f1xx_hal_gpio.c
stm32f1xx_hal_dma.c
stm32f1xx_hal_cortex.c
stm32f1xx_hal_pwr.c
stm32f1xx_hal_flash.c
stm32f1xx_hal_flash_ex.c
stm32f1xx_hal_exti.c
stm32f1xx_hal_rtc.c
stm32f1xx_hal_rtc_ex.c
stm32f1xx_hal_tim.c
stm32f1xx_hal_tim_ex.c
stm32f1xx_hal_uart.c
Drivers/CMSIS
system_stm32f1xx.c
Project Books Functions Templates
main.c
80 */
81 int main(void)
82 {
83 /* USER CODE BEGIN 1 */
84
85 /* USER CODE END 1 */
86
87
88 /* MCU Configuration--------------------------------------
89
90 /* Reset of all peripherals, Initializes the Flash interfac
91 HAL_Init();
92
93 /* USER CODE BEGIN Init */
94
95 /* USER CODE END Init */
96
97 /* Configure the system clock */
98 SystemClock_Config();
99
100 /* USER CODE BEGIN SysInit */
101
102 /* USER CODE END SysInit */
103
104 /* Initialize all configured peripherals */
105 MX_GPIO_Init();
106 MX_DMA_Init();
107 MX_ADC1_Init();
108 MX_TIM3_Init();
109 MX_USART1_UART_Init();
110 MX_RTC_Init();
111 /* USER CODE BEGIN 2 */
112
113 /* USER CODE END 2 */
114
Build Output
Program Size: Code=8476 RO-data=360 RW-data=16 ZI-data=1296
FromELF: creating hex file...
"PZ_D00\PZ_D00.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed: 00:00:41

生成的工程与代码
示意

```
369  static void MX_GPIO_Init(void)
370  {
371    GPIO_InitTypeDef GPIO_InitStruct = {0};
372
373    /* GPIO Ports Clock Enable */
374    __HAL_RCC_GPIOE_CLK_ENABLE();
375    __HAL_RCC_GPIOC_CLK_ENABLE();
376    __HAL_RCC_GPIOA_CLK_ENABLE();
377    __HAL_RCC_GPIOB_CLK_ENABLE();
378
379    /*Configure GPIO pin Output Level */
380    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_
381                             |GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6, GPIO_PIN
382
383    /*Configure GPIO pin Output Level */
384    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, GPIO_PIN_SET);
385
386    /*Configure GPIO pins : PE2 PE3 PE4 */
387    GPIO_InitStruct.Pin = GPIO_PIN_2|GPIO_PIN_3|GPIO_PIN_4;
388    GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
389    GPIO_InitStruct.Pull = GPIO_PULLUP;
390    HAL_GPIO_Init(GPIOE, &GPIO_InitStruct);
391
392    /*Configure GPIO pins : PC0 PC1 PC2 PC3
393                             PC4 PC5 PC6 */
394    GPIO_InitStruct.Pin = GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3
395                             |GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6;
396    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
397    GPIO_InitStruct.Pull = GPIO_NOPULL;
398    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
399    HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
400
401    /*Configure GPIO pin : PA0 */
402    GPIO_InitStruct.Pin = GPIO_PIN_0;
403    GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
404    GPIO_InitStruct.Pull = GPIO_PULLDOWN;
405    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
```

```
316    * @brief USART1 Initialization Function
317    * @param None
318    * @retval None
319    */
320  static void MX_USART1_UART_Init(void)
321  {
322
323    /* USER CODE BEGIN USART1_Init 0 */
324
325    /* USER CODE END USART1_Init 0 */
326
327    /* USER CODE BEGIN USART1_Init 1 */
328
329    /* USER CODE END USART1_Init 1 */
330    huart1.Instance = USART1;
331    huart1.Init.BaudRate = 9600;
332    huart1.Init.WordLength = UART_WORDLENGTH_8B;
333    huart1.Init.StopBits = UART_STOPBITS_1;
334    huart1.Init.Parity = UART_PARITY_NONE;
335    huart1.Init.Mode = UART_MODE_TX_RX;
336    huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
337    huart1.Init.OverSampling = UART_OVERSAMPLING_16;
338    if (HAL_UART_Init(&huart1) != HAL_OK)
339    {
340      Error_Handler();
341    }
342    /* USER CODE BEGIN USART1_Init 2 */
343
344    /* USER CODE END USART1_Init 2 */
345
346  }
347
348  /**
349    * Enable DMA controller clock
350    */
351  static void MX_DMA_Init(void)
352  {
353
354    /* DMA controller clock enable */
355    __HAL_RCC_DMA1_CLK_ENABLE();
356
```

生成的时钟相关代码
示意




```
175 void SystemClock_Config(void)
176 {
177   RCC_OscInitTypeDef RCC_OscInitStruct = {0};
178   RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
179   RCC_PeriphCLKInitTypeDef PeriphClkInit = {0};
180
181   /** Initializes the CPU, AHB and APB busses clocks
182   */
183   RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI|RCC_OSCILLATORTYPE_LSI;
184   RCC_OscInitStruct.HSIState = RCC_HSI_ON;
185   RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
186   RCC_OscInitStruct.LSIState = RCC_LSI_ON;
187   RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
188   RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI_DIV2;
189   RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL16;
190   if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
191   {
192     Error_Handler();
193   }
194   /** Initializes the CPU, AHB and APB busses clocks
195   */
196   RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
197                                 |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
198   RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
199   RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
200   RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
201   RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
202
203   if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
204   {
205     Error_Handler();
206   }
207   PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_RTC|RCC_PERIPHCLK_ADC;
208   PeriphClkInit.RTCClockSelection = RCC_RTCCLKSOURCE_LSI;
209   PeriphClkInit.AdcClockSelection = RCC_ADCPCLK2_DIV8;
210   if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK)
211   {
212     Error_Handler();
213   }
```

整理、小结

CubeMX 是可视化交互，可生成STM32 MCU工程文件及底层硬件相关代码的一种工具软件，为进行MCU应用的软、硬件开发、设计提供了良好支持。

其HAL思想值得关注

MCU 因包括较多硬件资源，对其使用往往需要参考技术资料，或利用库文件、或例程等办法。

在不同应用问题中，目的、要求各异、变化多样。对开发使用者有一定负担、困难，且容易出错。

```
#include "TIME_base.h"
void TIME_NVIC_Configuration(void)//如果系统会产生多种中断，那么就存在中断响应的优先级
{
    NVIC_InitTypeDef NVIC_InitStructure;
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);           //设置优先级分组
    NVIC_InitStructure.NVIC_IRQChannel = TIM6_IRQn;           //指定IRQ通道
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0; //指定先占优先级
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 3;        //从优先级
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;          //定义的IRQ是被使能还是失能
    NVIC_Init(&NVIC_InitStructure);
}

void TIME_Configuration(void)//配置TIM6
{
    TIM_TimeBaseInitTypeDef  TIM_TimeBaseStructure;
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM6, ENABLE);
    TIM_TimeBaseStructure.TIM_Period = 5000; //设置了在下一个更新事件装入活动的自动重装载寄存器周期的
    TIM_TimeBaseStructure.TIM_Prescaler =(7200-1); //设置了用来作为 TIMx 时钟频率除数的预分频值
    TIM_TimeBaseInit(TIM6, &TIM_TimeBaseStructure);
    TIM_ITConfig(TIM6,TIM_IT_Update|TIM_IT_Trigger,ENABLE);//使能或者失能 TIM 的中断，详见附录图1
    TIM_Cmd(TIM6, ENABLE);
}
```

2 一个汇编程序的观察

startup_stm32f103xe.s

```
33  Stack_Size      EQU     0x400
34
35                  AREA    STACK, NOINIT, READWRITE, ALIGN=3
36  Stack_Mem       SPACE   Stack_Size
37  __initial_sp
38
39  ; <h> Heap Configuration
40  ;   <o>  Heap Size (in Bytes) <0x0-0xFFFFFFFF:8>
41  ; </h>
42
43  Heap_Size       EQU     0x200
44
45                  AREA    HEAP, NOINIT, READWRITE, ALIGN=3
46  __heap_base
47  Heap_Mem        SPACE   Heap_Size
48  __heap_limit
49
50                  PRESERVE8
51                  THUMB
52
53
54  ; Vector Table Mapped to Address 0 at Reset
55                  AREA    RESET, DATA, READONLY
56                  EXPORT  __Vectors
57                  EXPORT  __Vectors_End
58                  EXPORT  __Vectors_Size
59
60  __Vectors       DCD     __initial_sp              ; Top of Stack
61                  DCD     Reset_Handler             ; Reset Handler
62                  DCD     NMI_Handler               ; NMI Handler
63                  DCD     HardFault_Handler         ; Hard Fault Handler
64                  DCD     MemManage_Handler         ; MPU Fault Handler
65                  DCD     BusFault_Handler          ; Bus Fault Handler
66                  DCD     UsageFault_Handler        ; Usage Fault Handler
```

```
                ENDP

                ALIGN

;*******************************************************
; 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

                ENDIF

                END
```

伪指令

- 在ARM汇编语言程序中，有一些特殊指令助记符，这些助记符与指令系统助记符不同，没有相对应的操作码。这些特殊指令助记符称为伪操作标识符，它们所完成的操作称为伪操作。伪操作是为了完成汇编程序做各种准备工作的，仅在汇编过程中起作用，一旦汇编结束，伪操作的使命就完成。

- 伪操作主要有符号定义伪操作、数据定义伪操作、汇编控制伪操作及其杂项伪操作等。

数据定义伪操作

用于为特定的数据分配存储单元，同时可完成已分配存储单元的初始化。常用的如下：

- DCB分配连续的字节存储单元并用指定的数据初始化
- DCW（DCWU）分配连续的半字存储单元并用指定数据初始化
- **DCD**（DCDU）分配连续的字存储单元并用指定数据初始化
- DCFD（DCFDU）为双精度浮点数分配连续字存储单元并用指定数据初始化
- DCFS（DCFSU）为单精度浮点数分配连续字存储单元并用指定数据初始化
- DCQ（DCQU）分配以8字节为单位的连续存储单元并用指定数据初始化
- SPACE分配连续的存储单元
- MAP定义一个结构化的内存表首地址
- FIELD定义一个结构化的内存表的数据域

```
60  __Vectors       DCD     __initial_sp              ; Top of Stack
61                  DCD     Reset_Handler             ; Reset Handler
62                  DCD     NMI_Handler               ; NMI Handler
63                  DCD     HardFault_Handler         ; Hard Fault Handler
64                  DCD     MemManage_Handler         ; MPU Fault Handler
65                  DCD     BusFault_Handler          ; Bus Fault Handler
66                  DCD     UsageFault_Handler        ; Usage Fault Handler
```

杂项伪操作

- AREA用于定义一个代码段或数据段。
- ALIGN用于使程序当前位置满足一定的对齐方式。
- ENTRY用于指定程序入口点。
- END用于指示源程序结束。
- EQU用于定义字符名称。
- EXPORT（或GLOBAL）用于声明符号可以被其他文件引用。
- IMPORT用于通知编译器当前符号不在本文件中。
- （10）GET（或INCLUDE）用于将一个文件包含到当前源文件。
- （11）INCBIN用于将一个文件包含到当前源文件。

```
Stack_Size      EQU     0x400

                AREA    STACK, NOINIT, READWRITE, ALIGN=3
Stack_Mem       SPACE   Stack_Size
__initial_sp

; <h> Heap Configuration
;   <o>  Heap Size (in Bytes) <0x0-0xFFFFFFFF:8>
; </h>

Heap_Size       EQU     0x200

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

                PRESERVE8
                THUMB


; Vector Table Mapped to Address 0 at Reset
                AREA    RESET, DATA, READONLY
                EXPORT  __Vectors
                EXPORT  __Vectors_End
                EXPORT  __Vectors_Size
```

汇编语言中的符号

符号可代替地址（addresses）、变量（variables）和常量（constants）等，以增加程序的灵活性和可读性。

- 符号区分大小写，同名的大、小写符号会被编译器认为是两个不同的符号。
- 符号在其作用范围内必须惟一。
- 自定义的符号名不能与系统的保留字相同。
- 符号名不应与指令或伪指令同名。
- 符号通常不能以数字开头。

```
                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
```

汇编语言的程序格式（参考前图）

- 在ARM汇编语言程序中以程序段为单位组织代码。段是相对独立的指令或数据序列，具有特定的名称。
- 可执行映像文件通常由以下几部分构成。
  - 一个或多个代码段，代码段的属性为只读。
  - 零个或多个数据段，数据段的属性为可读写。数据段可是被初始化的数据段或没有被初始化的数据段。
- 链接器根据系统默认或用户设定的规则，将各个段安排在存储器中的相应位置。

汇编程序示例

C语言程序，该程序实现了著名的Euclid最大公约数算法。

```
int gcd(int a, int b)
{
    while (a != b)
     {
       if (a > b)
            a = a - b;
       else
            b = b - a;
     }
    return a;
}
```

用ARM汇编语言重写这个例子，如下所示【程序1】

```
gcd   CMP     r0, r1
      BEQ     end
      BLT     l ess
      SUB     r0, r0, r1
      B       gcd
less
      SUB     r1, r1, r0
      B       gcd
End
```

充分地利用条件执行修改左面的例子，得到【程序2】。

```
gcd
      CMP     r0, r1
      SUBGT   r0, r0, r1
      SUBLT   r1, r1, r0
      BNE     gcd
```

【程序1】仅使用了分支指令，【程序2】充分利用了ARM指令条件执行的特点，仅使用了4条指令就完成了全部算法。这对提供程序的代码密度和执行速度十分有帮助。分支指令十分影响处理器的速度。每次执行分支指令，处理器都会排空流水线，重新装载指令。

C程序编译
实际代码观察

3个分支语句

```
0x080021CC E003      B        0x080021D6
    77:          {       if (a > b)     a = a - b;
0x080021CE DD01      BLE      0x080021D4
0x080021D0 1A40      SUBS     r0,r0,r1
0x080021D2 E000      B        0x080021D6
    78:                  else            b = b - a;
    79:          }
    80:          return a;
0x080021D4 1A09      SUBS     r1,r1,r0
    76:          while (a != b)
    77:          {       if (a > b)     a = a - b;
    78:                  else            b = b - a;
    79:          }
    80:          return a;
0x080021D6 4288      CMP      r0,r1
0x080021D8 D1F9      BNE      0x080021CE
    81: }
    82:
    83: /* USER CODE END 0 */
    84:
    85: /**
```

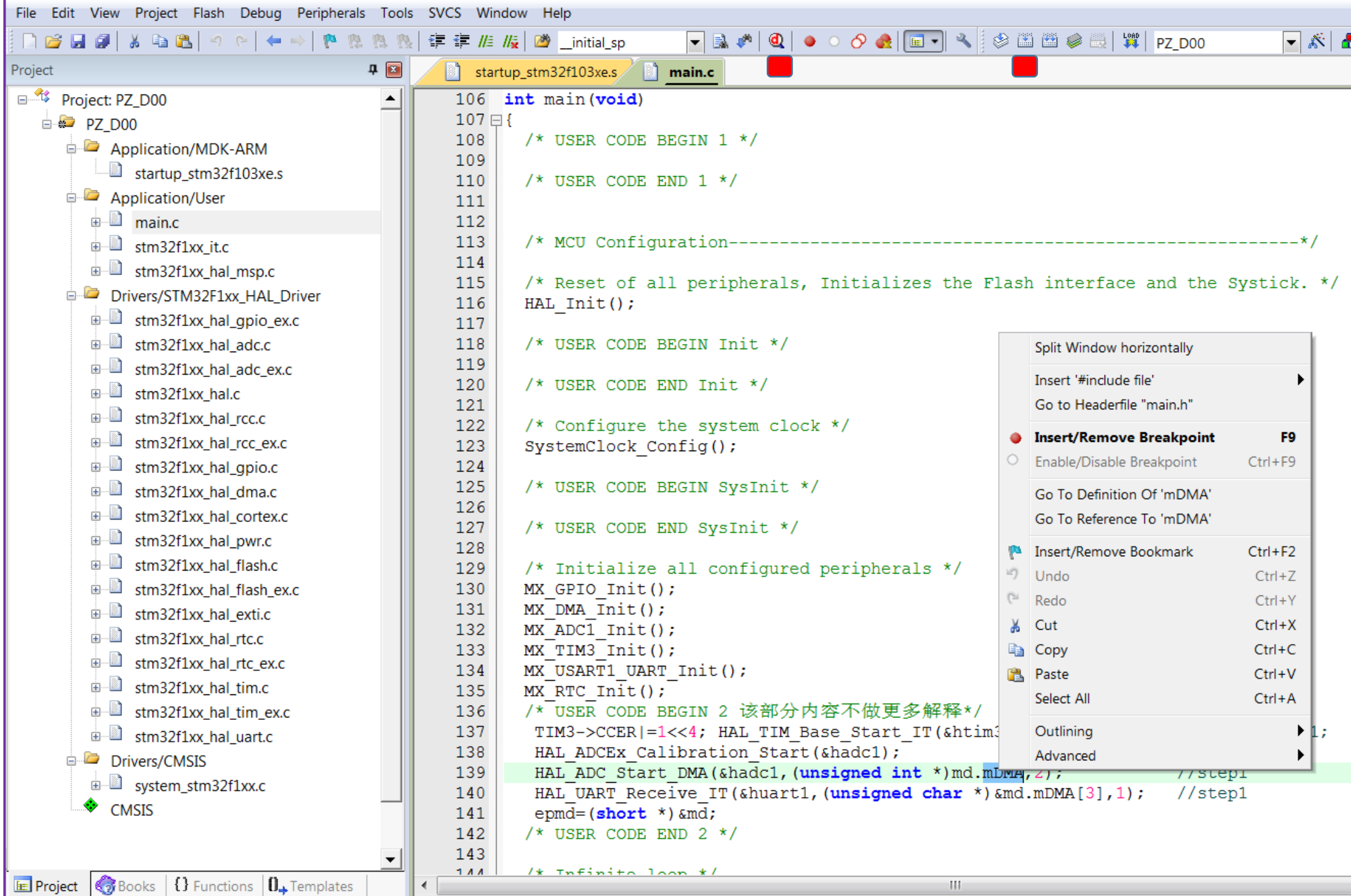
startup_stm32f103xe.s | main.c | core_cm3.h

```
73  /* USER CODE BEGIN 0 */
74  int gcd(int a, int b)
75  {
76     while (a != b)
77     {  if (a > b)  a = a - b;
78        else        b = b - a;
79     }
80     return a;
81  }
```

3 MCU 软件开发工具

Keil

（更多使用内容需自行学习了解）



main.c | startup_stm32f103xe.s | stm32f1xx.h | **PZ_D00.map**

3 编译的结果文件（map）

```
1210     _microlib_exit                           - Undefined Weak Reference
1211     __Vectors_Size                  0x00000130   Number         0  startup_stm32f103xe.o ABSOLUTE
1212     __Vectors                       0x08000000   Data           4  startup_stm32f103xe.o(RESET)
1213     __Vectors_End                   0x08000130   Data           0  startup_stm32f103xe.o(RESET)
1214     __main                          0x08000131   Thumb Code     0  entry.o(.ARM.Collect$$$$00000000)
1215     _main_stk                       0x08000131   Thumb Code     0  entry2.o(.ARM.Collect$$$$00000001)
1216     _main_scatterload               0x08000135   Thumb Code     0  entry5.o(.ARM.Collect$$$$00000004)
1217     __main_after_scatterload        0x08000139   Thumb Code     0  entry5.o(.ARM.Collect$$$$00000004)
1218     _main_clock                     0x08000139   Thumb Code     0  entry7b.o(.ARM.Collect$$$$00000008)
1219     _main_cpp_init                  0x08000139   Thumb Code     0  entry8b.o(.ARM.Collect$$$$0000000A)
1220     _main_init                      0x08000139   Thumb Code     0  entry9a.o(.ARM.Collect$$$$0000000B)
1221     __rt_final_cpp                  0x08000141   Thumb Code     0  entry10a.o(.ARM.Collect$$$$0000000D)
1222     __rt_final_exit                 0x08000141   Thumb Code     0  entry11a.o(.ARM.Collect$$$$0000000F)
1223     Reset_Handler                   0x08000145   Thumb Code     8  startup_stm32f103xe.o(.text)
1224     ADC1_2_IRQHandler               0x0800015f   Thumb Code     0  startup_stm32f103xe.o(.text)
1225     ADC3_IRQHandler                 0x0800015f   Thumb Code     0  startup_stm32f103xe.o(.text)
1226     CAN1_RX1_IRQHandler             0x0800015f   Thumb Code     0  startup_stm32f103xe.o(.text)
1227     CAN1_SCE_IRQHandler             0x0800015f   Thumb Code     0  startup_stm32f103xe.o(.text)
1228     DMA1_Channel2_IRQHandler        0x0800015f   Thumb Code     0  startup_stm32f103xe.o(.text)
1229     DMA1_Channel3_IRQHandler        0x0800015f   Thumb Code     0  startup_stm32f103xe.o(.text)
1230     DMA1_Channel4_IRQHandler        0x0800015f   Thumb Code     0  startup_stm32f103xe.o(.text)
1231     DMA1_Channel5_IRQHandler        0x0800015f   Thumb Code     0  startup_stm32f103xe.o(.text)
1232     DMA1_Channel6_IRQHandler        0x0800015f   Thumb Code     0  startup_stm32f103xe.o(.text)
1233     DMA1_Channel7_IRQHandler        0x0800015f   Thumb Code     0  startup_stm32f103xe.o(.text)
1234     DMA2_Channel1_IRQHandler        0x0800015f   Thumb Code     0  startup_stm32f103xe.o(.text)
1235     DMA2_Channel2_IRQHandler        0x0800015f   Thumb Code     0  startup_stm32f103xe.o(.text)
1236     DMA2_Channel3_IRQHandler        0x0800015f   Thumb Code     0  startup_stm32f103xe.o(.text)
1237     DMA2_Channel4_5_IRQHandler      0x0800015f   Thumb Code     0  startup_stm32f103xe.o(.text)
1238     EXTI0_IRQHandler                0x0800015f   Thumb Code     0  startup_stm32f103xe.o(.text)
1239     EXTI15_10_IRQHandler            0x0800015f   Thumb Code     0  startup_stm32f103xe.o(.text)
1240     EXTI1_IRQHandler                0x0800015f   Thumb Code     0  startup_stm32f103xe.o(.text)
1241     EXTI2_IRQHandler                0x0800015f   Thumb Code     0  startup_stm32f103xe.o(.text)
1242     EXTI3_IRQHandler                0x0800015f   Thumb Code     0  startup_stm32f103xe.o(.text)
1243     EXTI4_IRQHandler                0x0800015f   Thumb Code     0  startup_stm32f103xe.o(.text)
1244     EXTI9_5_IRQHandler              0x0800015f   Thumb Code     0  startup_stm32f103xe.o(.text)
1245     FLASH_IRQHandler                0x0800015f   Thumb Code     0  startup_stm32f103xe.o(.text)
```

3 编译的结果文件（map）

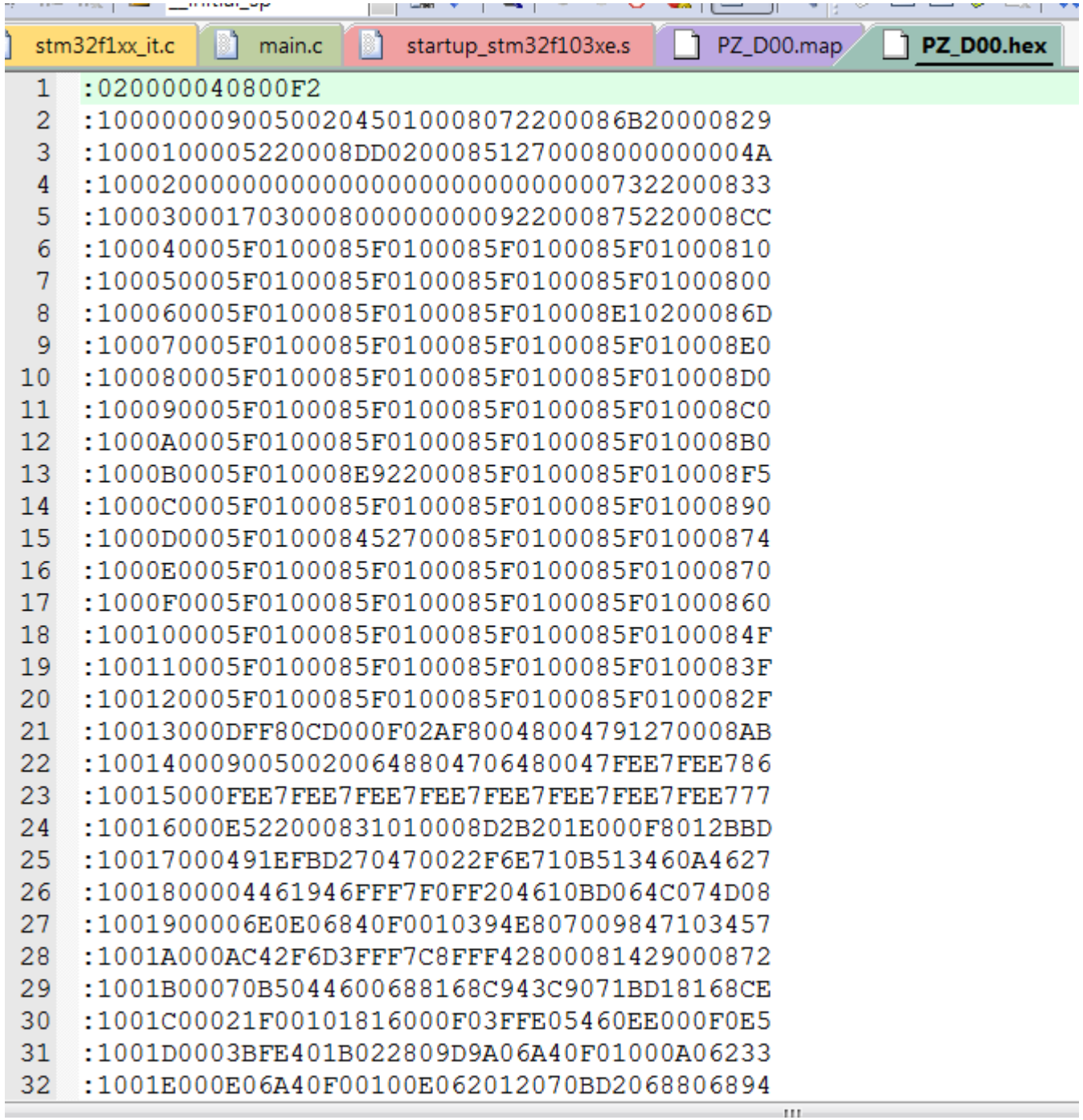
stm32f1xx_it.c | main.c | startup_stm32f103xe.s | **PZ_D00.map**

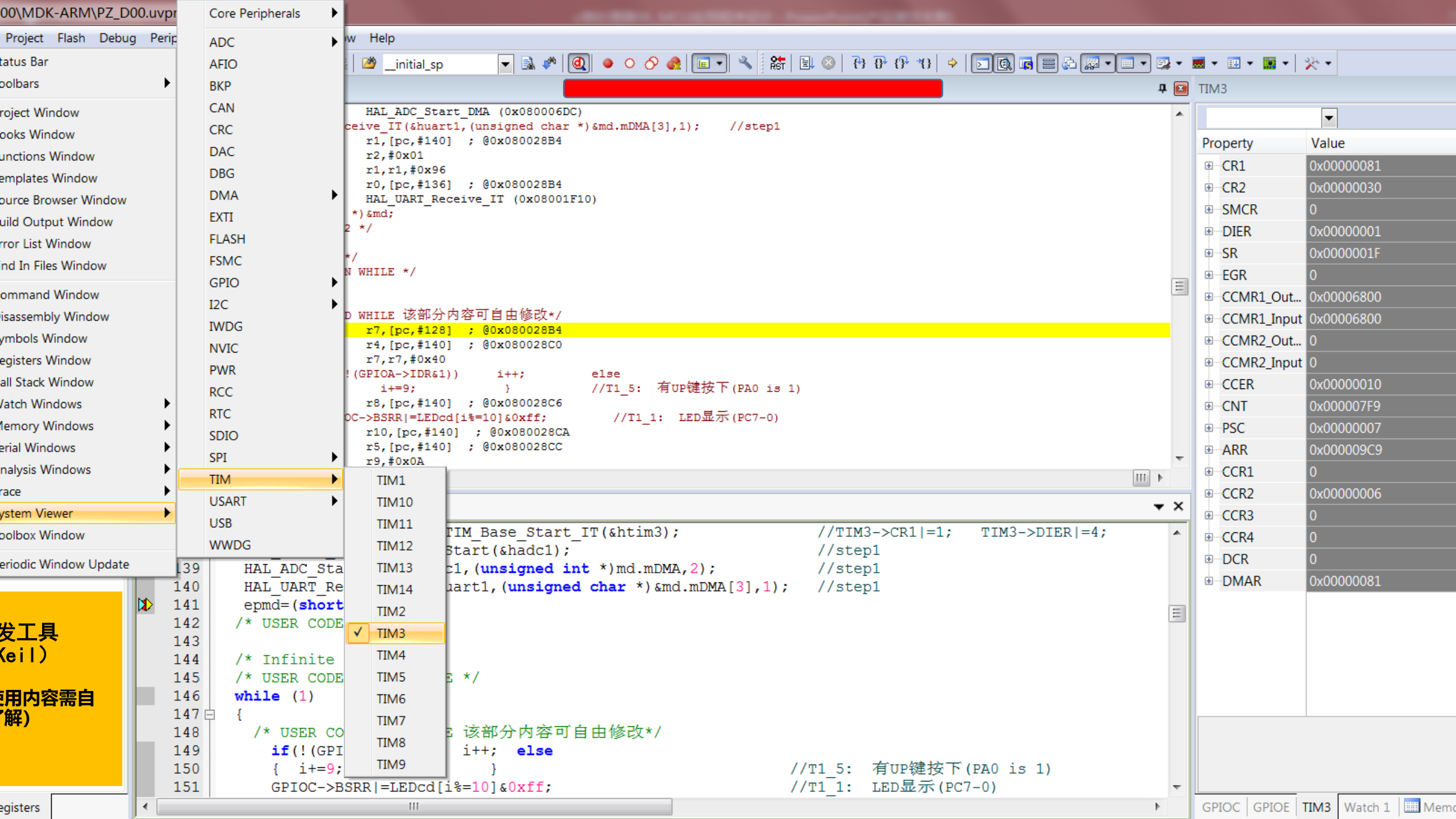
```
1364    TIM3_IRQHandler                  0x080022e9   Thumb Code     6  stm32f1xx_it.o(i.TIM3_IRQHandler)
1365    TIM_Base_SetConfig               0x080022f5   Thumb Code   120  stm32f1xx_hal_tim.o(i.TIM_Base_SetConfig)
1366    TIM_ETR_SetConfig                0x08002381   Thumb Code    20  stm32f1xx_hal_tim.o(i.TIM_ETR_SetConfig)
1367    TIM_OC2_SetConfig                0x08002405   Thumb Code    98  stm32f1xx_hal_tim.o(i.TIM_OC2_SetConfig)
1368    USART1_IRQHandler                0x08002745   Thumb Code     6  stm32f1xx_it.o(i.USART1_IRQHandler)
1369    UsageFault_Handler               0x08002751   Thumb Code     2  stm32f1xx_it.o(i.UsageFault_Handler)
1370    __scatterload_copy               0x08002773   Thumb Code    14  handlers.o(i.__scatterload_copy)
1371    __scatterload_null               0x08002781   Thumb Code     2  handlers.o(i.__scatterload_null)
1372    __scatterload_zeroinit           0x08002783   Thumb Code    14  handlers.o(i.__scatterload_zeroinit)
1373    main                             0x08002791   Thumb Code   282  main.o(i.main)
1374    LEDcd                            0x080028d0   Data          10  main.o(.constdata)
1375    AHBPrescTable                    0x080028da   Data          16  system_stm32f1xx.o(.constdata)
1376    APBPrescTable                    0x080028ea   Data           8  system_stm32f1xx.o(.constdata)
1377    Region$$Table$$Base              0x080028f4   Number         0  anon$$obj.o(Region$$Table)
1378    Region$$Table$$Limit             0x08002914   Number         0  anon$$obj.o(Region$$Table)
1379    i                                0x20000000   Data           2  main.o(.data)
1380    x                                0x20000002   Data           2  main.o(.data)
1381    epmd                             0x20000004   Data           4  main.o(.data)
1382    uwTickFreq                       0x20000008   Data           1  stm32f1xx_hal.o(.data)
1383    uwTickPrio                       0x2000000c   Data           4  stm32f1xx_hal.o(.data)
1384    uwTick                           0x20000010   Data           4  stm32f1xx_hal.o(.data)
1385    SystemCoreClock                  0x20000014   Data           4  system_stm32f1xx.o(.data)
1386    hadc1                            0x20000018   Data          48  main.o(.bss)
1387    hrtc                             0x20000048   Data          20  main.o(.bss)
1388    htim3                            0x2000005c   Data          72  main.o(.bss)
1389    huart1                           0x200000a4   Data          64  main.o(.bss)
1390    md                               0x200000e4   Data         104  main.o(.bss)
1391    hdma_adc1                        0x2000014c   Data          68  main.o(.bss)
1392    __initial_sp                     0x20000590   Data           0  startup_stm32f103xe.o(STACK)
1393
```

3 编译的结果文件（map）

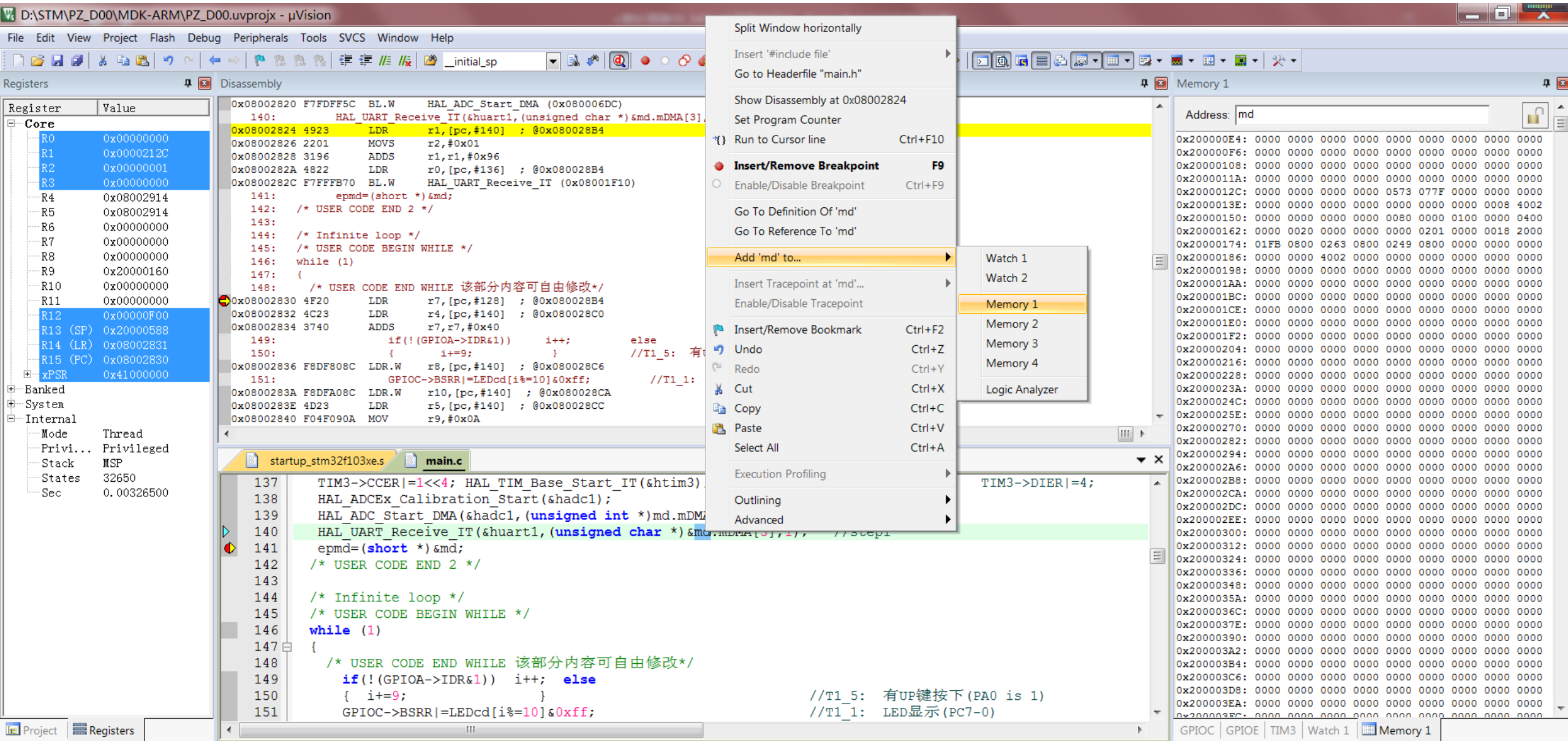
```
1568
1569      Code (inc. data)    RO Data    RW Data    ZI Data      Debug   Object Name
1570
1571      1042         88         10          8        376     517820   main.o
1572        36          8        304          0       1024        764   startup_stm32f103xe.o
1573       128         24          0         12          0       5633   stm32f1xx_hal.o
1574      1114         60          0          0          0       8190   stm32f1xx_hal_adc.o
1575       200          6          0          0          0       1366   stm32f1xx_hal_adc_ex.o
1576       198         14          0          0          0      28671   stm32f1xx_hal_cortex.o
1577      1166         52          0          0          0       4222   stm32f1xx_hal_dma.o
1578       506         42          0          0          0       2744   stm32f1xx_hal_gpio.o
1579       540         80          0          0          0       3870   stm32f1xx_hal_msp.o
1580        12          4          0          0          0        475   stm32f1xx_hal_pwr.o
1581      1276        100          0          0          0       5112   stm32f1xx_hal_rcc.o
1582       444         44          0          0          0       2533   stm32f1xx_hal_rcc_ex.o
1583       322          6          0          0          0       3540   stm32f1xx_hal_rtc.o
1584      1704         76          0          0          0      14962   stm32f1xx_hal_tim.o
1585       128         20          0          0          0       2301   stm32f1xx_hal_tim_ex.o
1586      1174         10          0          0          0       8543   stm32f1xx_hal_uart.o
1587       220         38          0          2          0       4775   stm32f1xx_it.o
1588         2          0         24          4          0        979   system_stm32f1xx.o
1589
1590    ----------------------------------------------------------------------
1591     10238        672        372         28       1404     616500   Object Totals
1592         0          0         32          0          0          0   (incl. Generated)
1593        26          0          2          2          4          0   (incl. Padding)
1594
1595    ----------------------------------------------------------------------
1596
1597      Code (inc. data)    RO Data    RW Data    ZI Data      Debug   Library Member Name
1598
1599         0          0          0          0          0          0   entry.o
1600         0          0          0          0          0          0   entry10a.o
1601         0          0          0          0          0          0   entry11a.o
1602         8          4          0          0          0          0   entry2.o
1603         4          0          0          0          0          0   entry5.o
1604         0          0          0          0          0          0   entry7b.o
1605         0          0          0          0          0          0   entry8b.o
1606         8          4          0          0          0          0   entry9a.o
```

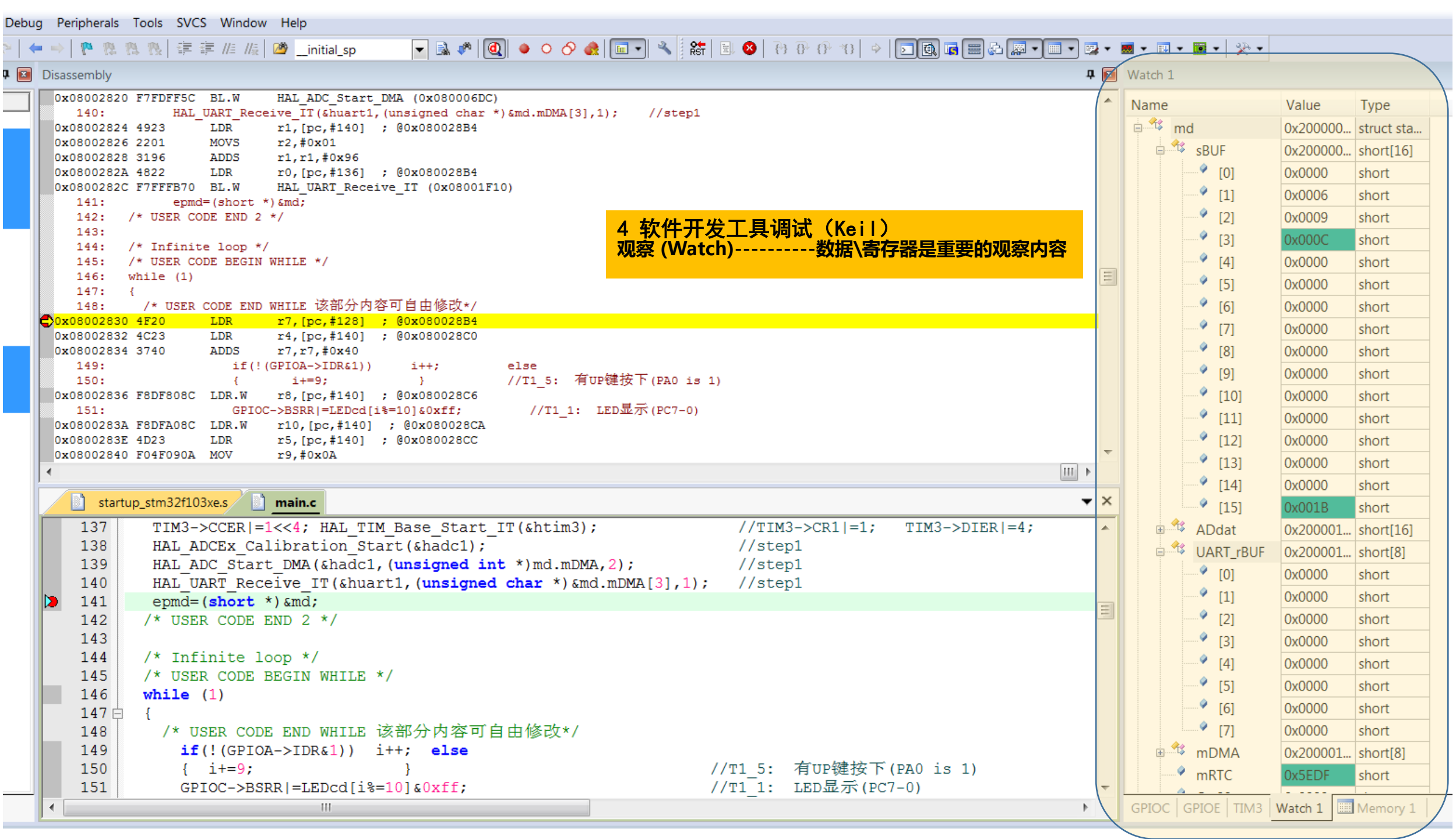
3 编译的结果文件（Hex）




Core Peripherals
ADC
AFIO
BKP
CAN
CRC
DAC
DBG
DMA
EXTI
FLASH
FSMC
GPIO
I2C
IWDG
NVIC
PWR
RCC
RTC
SDIO
SPI
TIM
USART
USB
WWDG
TIM1
TIM10
TIM11
TIM12
TIM13
TIM14
TIM2
TIM3
TIM4
TIM5
TIM6
TIM7
TIM8
TIM9
Project
Flash
Debug
Help
TIM3
Property
Value
CR1
CR2
SMCR
DIER
SR
EGR
CCMR1_Out...
CCMR1_Input
CCMR2_Out...
CCMR2_Input
CCER
CNT
PSC
ARR
CCR1
CCR2
CCR3
CCR4
DCR
DMAR
GPIOC
GPIOE
TIM3
Watch 1

4 MCU
软件开发工具调试（Keil）
观察存贮器 （Memory)----------数据是重要的观察内容




Debug Peripherals Tools SVCS Window Help
_initial_sp
Disassembly
0x08002820 F7FDFF5C BL.W HAL_ADC_Start_DMA (0x080006DC)
140: HAL_UART_Receive_IT(&huart1,(unsigned char *)&md.mDMA[3],1); //step1
0x08002824 4923 LDR r1,[pc,#140] ; @0x080028B4
0x08002826 2201 MOVS r2,#0x01
0x08002828 3196 ADDS r1,r1,#0x96
0x0800282A 4822 LDR r0,[pc,#136] ; @0x080028B4
0x0800282C F7FFFB70 BL.W HAL_UART_Receive_IT (0x08001F10)
141: epmd=(short *)&md;
142: /* USER CODE END 2 */
143:
144: /* Infinite loop */
145: /* USER CODE BEGIN WHILE */
146: while (1)
147: {
148: /* USER CODE END WHILE 该部分内容可自由修改*/
0x08002830 4F20 LDR r7,[pc,#128] ; @0x080028B4
0x08002832 4C23 LDR r4,[pc,#140] ; @0x080028C0
0x08002834 3740 ADDS r7,r7,#0x40
149: if(!(GPIOA->IDR&1)) i++; else
150: { i+=9; } //T1_5: 有UP键按下(PA0 is 1)
0x08002836 F8DF808C LDR.W r8,[pc,#140] ; @0x080028C6
151: GPIOC->BSRR|=LEDcd[i%=10]&0xff; //T1_1: LED显示(PC7-0)
0x0800283A F8DFA08C LDR.W r10,[pc,#140] ; @0x080028CA
0x0800283E 4D23 LDR r5,[pc,#140] ; @0x080028CC
0x08002840 F04F090A MOV r9,#0x0A
4 软件开发工具调试（Keil）
观察 (Watch)----------数据\寄存器是重要的观察内容
startup_stm32f103xe.s
main.c
137 TIM3->CCER|=1<<4; HAL_TIM_Base_Start_IT(&htim3); //TIM3->CR1|=1; TIM3->DIER|=4;
138 HAL_ADCEx_Calibration_Start(&hadc1); //step1
139 HAL_ADC_Start_DMA(&hadc1,(unsigned int *)md.mDMA,2); //step1
140 HAL_UART_Receive_IT(&huart1,(unsigned char *)&md.mDMA[3],1); //step1
141 epmd=(short *)&md;
142 /* USER CODE END 2 */
143
144 /* Infinite loop */
145 /* USER CODE BEGIN WHILE */
146 while (1)
147 {
148 /* USER CODE END WHILE 该部分内容可自由修改*/
149 if(!(GPIOA->IDR&1)) i++; else
150 { i+=9; } //T1_5: 有UP键按下(PA0 is 1)
151 GPIOC->BSRR|=LEDcd[i%=10]&0xff; //T1_1: LED显示(PC7-0)
Watch 1
Name Value Type
md 0x200000... struct sta...
sBUF 0x200000... short[16]
[0] 0x0000 short
[1] 0x0006 short
[2] 0x0009 short
[3] 0x000C short
[4] 0x0000 short
[5] 0x0000 short
[6] 0x0000 short
[7] 0x0000 short
[8] 0x0000 short
[9] 0x0000 short
[10] 0x0000 short
[11] 0x0000 short
[12] 0x0000 short
[13] 0x0000 short
[14] 0x0000 short
[15] 0x001B short
ADdat 0x200001... short[16]
UART_rBUF 0x200001... short[8]
[0] 0x0000 short
[1] 0x0000 short
[2] 0x0000 short
[3] 0x0000 short
[4] 0x0000 short
[5] 0x0000 short
[6] 0x0000 short
[7] 0x0000 short
mDMA 0x200001... short[8]
mRTC 0x5EDF short
GPIOC GPIOE TIM3 Watch 1 Memory 1

5　微处理器程序设计参考代码

startup_stm32f103xe.s　main.c

```
143
144   /* Infinite loop */
145   /* USER CODE BEGIN WHILE */
146   while (1)
147   {
148     /* USER CODE END WHILE 该部分内容可自由修改*/
149       if(!(GPIOA->IDR&1))  i++;  else
150       {  i+=9;                  }                                  //T1_5:  有UP键按下(PA0 is 1)
151       GPIOC->BSRR|=LEDcd[i%=10]&0xff;                              //T1_1:  LED显示(PC7-0)
152       Delay(500+i*23);         GPIOC->BRR|=0xff;                   //T1_2:  延迟，  0.5s+学号个位数*0.1s
153 //-------------------------------------
154       md.sBUF[i]=i*3+x;                                            //T1_3:  填充学号各位至sBUF[0--x]
155       if(i==0) {  md.sBUF[14]=md.sBUF[15];   md.sBUF[15]=0;}
156       md.sBUF[15]+=md.sBUF[i];                                     //T1_4:  累加sBUF[x]，最终累加和存至sBUF[
157 //自行扩展Begin
158
159
160
161 //自行扩展END
162   }
```

微处理器程序设计参考代码

startup_stm32f103xe.s | main.c

```
74  const char LEDcd[10]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};
75  short i=0,x=3,*epmd;
76  void Delay(unsigned tDly)
77  {  short tDi;
78     while(tDly--)
79     {  tDi=4000;
80        while(tDi--)
81        {  if(!(md.mDMA[6]&0x10))  continue;
82                md.mDMA[6]&=~0x10;    md.mDMA[5]++;
83                                                    //can do some works, but no stay here more time
84        }
85  //自行扩展Begin......u can do something
86        md.mRTC=RTC->CNTL;
87
88
89  //自行扩展END  ......u can do something
90     };
91  }
92  //---------????------------------------------------------------
93  void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
94  {  md.mDMA[2]++;  md.mDMA[2]&=0x7;
95     md.UART_rBUF[md.mDMA[2]]=md.mDMA[3];
96     HAL_UART_Receive_IT(&huart1,(unsigned char *)&md.mDMA[3],1);    // USART1->DR=md.mDMA[3]+1;
97     md.mDMA[4]=md.mDMA[3]+1;
98     while(HAL_UART_Transmit(&huart1,(unsigned char *)&md.mDMA[4],1,5000)!=HAL_OK);
99  }
```

微处理器程序设计参考代码-----定时器TIM3中断服务（ISR）程序

startup_stm32f103xe.s | main.c | **stm32f1xx_it.c**

```
217   * @brief This function handles TIM3 global interrupt.
218   */
219 extern short *epmd;
220 void TIM3_IRQHandler(void)
221 {
222   /* USER CODE BEGIN TIM3_IRQn 0  该部分内容可自由修改 */
223 static unsigned short LEDpwm;
224    LEDpwm++;   LEDpwm%=9800;
225    if(LEDpwm<4900)   TIM3->CCR2=       LEDpwm/2+88;
226    else              TIM3->CCR2=4900-  LEDpwm/2+88;
227    if(LEDpwm%10==1)  {  epmd[47]++;    epmd[46]|=0x000f;}
228    if(LEDpwm%100==0)                   epmd[46]|=0x00f0;            //Set a Click Flag
229
230    if(!(GPIOE->IDR&0x08))                                           //PE2(down Key) push
231    {  if(LEDpwm%8>3) GPIOB->BSRR|=1<<5;   else  GPIOB->BRR|=1<<5; } //Beep
232 //ISR 自行扩展Begin
233
234
235
236 //ISR 自行扩展END
237    /* USER CODE END TIM3_IRQn 0 */
238   HAL_TIM_IRQHandler(&htim3);
239   /* USER CODE BEGIN TIM3_IRQn 1 */
240
241   /* USER CODE END TIM3_IRQn 1 */
242 }
```

微处理器程序设计参考代码-----定时器TIM3中断服务（ISR）程序

代码变动比较

```
217 ··*·@brief·This·function·handles·TIM3·global·interrupt.↵
218 ··*/↵
219 void·TIM3_IRQHandler(void)↵
220 {↵
221 ··/*·USER·CODE·BEGIN·TIM3_IRQn·0·*/↵
222 ↵
223 ··/*·USER·CODE·END·TIM3_IRQn·0·*/↵
224 ··HAL_TIM_IRQHandler(&htim3);↵
225 ··/*·USER·CODE·BEGIN·TIM3_IRQn·1·*/↵
```

```
217 ··*·@brief·This·function·handles·TIM3·global·interrupt.↵
218 ··*/↵
+219 extern·short·*epmd;↵
220 void·TIM3_IRQHandler(void)↵
221 {↵
+222 ··/*·USER·CODE·BEGIN·TIM3_IRQn·0··该部分内容可自由修改·*/↵
+223 static·unsigned·short·LEDpwm;↵
+224 ——→LEDpwm++;——→LEDpwm%=9800;↵
+225 ——→if(LEDpwm<4900)→TIM3->CCR2=→——→——→LEDpwm/2+88;↵
+226 ——→else——→——→——→——→——→TIM3->CCR2=4900-——→LEDpwm/2+88;↵
+227 ——→if(LEDpwm%10==1)——→{——→epmd[47]++;→——→epmd[46]|=0x000f;}↵
+228 ——→if(LEDpwm%100==0)——→——→——→——→——→——→——→epmd[46]|=0x00f0;——→——→——→——→——→
229 ↵
+230 ——→if(!(GPIOE->IDR&0x08))——→——→——→——→——→——→——→——→——→——→——→——→——→——→——→——→
+231 ——→{——→if(LEDpwm%8>3)→GPIOB->BSRR|=1<<5;→else·——→GPIOB->BRR|=1<<5;——→}——→//Beep↵
+232 //ISR·自行扩展Begin↵
233 ——→——→↵
234 ——→——→↵
235 ——→——→↵
+236 //ISR·自行扩展END↵
+237 ——→/*·USER·CODE·END·TIM3_IRQn·0·*/↵
238 ··HAL_TIM_IRQHandler(&htim3);↵
239 ··/*·USER·CODE·BEGIN·TIM3_IRQn·1·*/↵
```