

Microprocessor – Microcontroller

201 W2.2 - Embedded C Programming

Nguyen Tran Huu Nguyen

D: Computer Engineering

E: nthnguyen@hcmut.edu.vn



A simple C program

```
void FSM();

void main(void){
    // initialize the device
    System_Initialization();
    while (1) {
        FSM();
    }
}

/**
 * @brief This function handles TIM interrupt
 * request.
 * @param None
 * @retval None */
void TIM3_IRQHandler(void){
    HAL_TIM_IRQHandler(&TimHandle);
}
```



```
/**
 * @brief Period elapsed callback in non blocking
 * mode
 * @param htim : TIM handle
 * @retval None */
void
HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef
*htim) {
    SCH_Update();
}
```

A main() Function

- System initialization
 - Oscillator
 - Input/Output
 - Special peripherals/modules: Timers, CCP, ADC
 - Disable/Enable interrupt
- A super loop while(1)
- One or more finite state machine (FSM)

System initialization

```
35 enum InitState initState = HAL_INIT;
36
37 void System Initialization(void)
38 {
39     while(initState != MAX_INIT_STATE){
40         switch (initState) {
41             case HAL_INIT:
42                 HAL_Init();
43                 break;
44             case SYSTEM_CLOCK_INIT:
45                 SystemClock_Config();
46                 break;
47             case UART_INIT:
48                 UART3_Init();
49                 UART1_Init();
50                 DEBUG_INIT(UART3_SendToHost((uint8_t*)"UART_INIT - Done \r\n"));
51                 break;
52             case GPIO_INIT:
53                 MX_GPIO_Init();
54                 DEBUG_INIT(UART3_SendToHost((uint8_t*)"GPIO_INIT - ADC_DMA_Init - Done \r\n"));
55                 break;
56             case LED_DISPLAY_INIT:
57                 Led_Display_Init();
58                 DEBUG_INIT(UART3_SendToHost((uint8_t*)"LED_DISPLAY_INIT - Done \r\n"));
59                 break;
60             case RELAY_INIT:
61                 Relay_Init();
62                 DEBUG_INIT(UART3_SendToHost((uint8_t*)"RELAY_INIT - Done \r\n"));
63                 break;
64             case FLASH_INIT:
65                 DEBUG_INIT(UART3_SendToHost((uint8_t*)"FLASH_INIT - Done \r\n"));
66                 break;
67             case TIMER_INIT:
68                 Timer_Init();
69                 DEBUG_INIT(UART3_SendToHost((uint8_t*)"TIMER_INIT - Done \r\n"));
70                 break;
71             case SPI_INIT:
72                 SPI1_Init();
73                 SPI2_Init();
74                 DEBUG_INIT(UART3_SendToHost((uint8_t*)"SPI_INIT - Done \r\n"));
75                 break;
76             case SPI_25LCXXX_INIT:
77                 Eeprom_Initialize();
78                 DEBUG_INIT(UART3_SendToHost((uint8_t*)"SPI_25LCXXX_INIT - Done \r\n"));
79                 break;
80             case I2C_INIT:
81                 I2C_Init();
82                 DEBUG_INIT(UART3_SendToHost((uint8_t*)"I2C_Init \r\n"));
83                 PCF_Init();
```

HAL_Init

This function is used to initialize the HAL Library; it must be the first instruction to be executed in the main program (before to call any other HAL function), it performs the following:

- * Configure the Flash prefetch.
- * Configures the SysTick to generate an interrupt each 1 millisecond, which is clocked by the HSI (at this stage, the clock is not yet configured and thus the system is running from the internal HSI at 16 MHz).
- * Set NVIC Group Priority to 4.
- * Calls the HAL_MspInit() callback function defined in user file "**stm32f1xx_hal_msp.c**" to do the global low level hardware initialization
- * @note SysTick is used as time base for the HAL_Delay() function, the application need to ensure that the SysTick time base is always set to 1 millisecond to have correct HAL operation.
- * @retval HAL status

HAL_Init

```
142 HAL_StatusTypeDef HAL_Init(void)
143 {
144     /* Configure Flash prefetch */
145     #if (PREFETCH_ENABLE != 0)
146     #if defined(STM32F101x6) || defined(STM32F101xB) || defined(STM32F101xE) || defined(STM32F101xG) || \
147         defined(STM32F102x6) || defined(STM32F102xB) || \
148         defined(STM32F103x6) || defined(STM32F103xB) || defined(STM32F103xE) || defined(STM32F103xG) || \
149         defined(STM32F105xC) || defined(STM32F107xC)
150
151         /* Prefetch buffer is not available on value line devices */
152         __HAL_FLASH_PREFETCH_BUFFER_ENABLE();
153     #endif
154     #endif /* PREFETCH_ENABLE */
155
156     /* Set Interrupt Group Priority */
157     HAL_NVIC_SetPriorityGrouping(NVIC_PRIORITYGROUP_4);
158
159     /* Use systick as time base source and configure 1ms tick (default clock after Reset is HSI) */
160     HAL_InitTick(TICK_INT_PRIORITY);
161
162     /* Init the low level hardware */
163     HAL_MspInit();
164
165     /* Return function status */
166     return HAL_OK;
167 }
```

SystemClock_Config

```
10 /**
11  * @brief System Clock Configuration
12  * @retval None
13  */
14 void SystemClock_Config(void)
15 {
16     RCC_OscInitTypeDef RCC_OscInitStruct = {0};
17     RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
18     RCC_PeriphCLKInitTypeDef PeriphClkInit = {0};
19     /** Initializes the CPU, AHB and APB busses clocks
20     */
21     RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI|RCC_OSCILLATORTYPE_LSI;
22     RCC_OscInitStruct.HSEState      = RCC_HSE_OFF;
23     RCC_OscInitStruct.LSEState      = RCC_LSE_OFF;
24     RCC_OscInitStruct.HSIState      = RCC_HSI_ON;
25     RCC_OscInitStruct.LSIState      = RCC_LSI_ON;
26
27     RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
28     RCC_OscInitStruct.HSEPredivValue     = RCC_HSE_PREDIV_DIV1;
29     RCC_OscInitStruct.PLL.PLLState      = RCC_PLL_ON;
30     RCC_OscInitStruct.PLL.PLLSource     = RCC_PLLSOURCE_HSI_DIV2;
31     RCC_OscInitStruct.PLL.PLLMUL       = RCC_PLL_MUL16;
32     if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK){
33         Error_Handler();
34     }
35     /** Initializes the CPU, AHB and APB busses clocks
36     */
37     RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
38                                 |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
39     RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
40     RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
41     RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
42     RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
43
44     if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK){
45         Error_Handler();
46     }
47     PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_ADC;
48     PeriphClkInit.AdcClockSelection = RCC_ADCPCLK2_DIV6;
49     if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK){
50         Error_Handler();
51     }
52 }
53
54
```


Clocks

Three different clock sources can be used to drive the system clock (SYSCLK):

- HSI oscillator clock
- HSE oscillator clock
- PLL clock

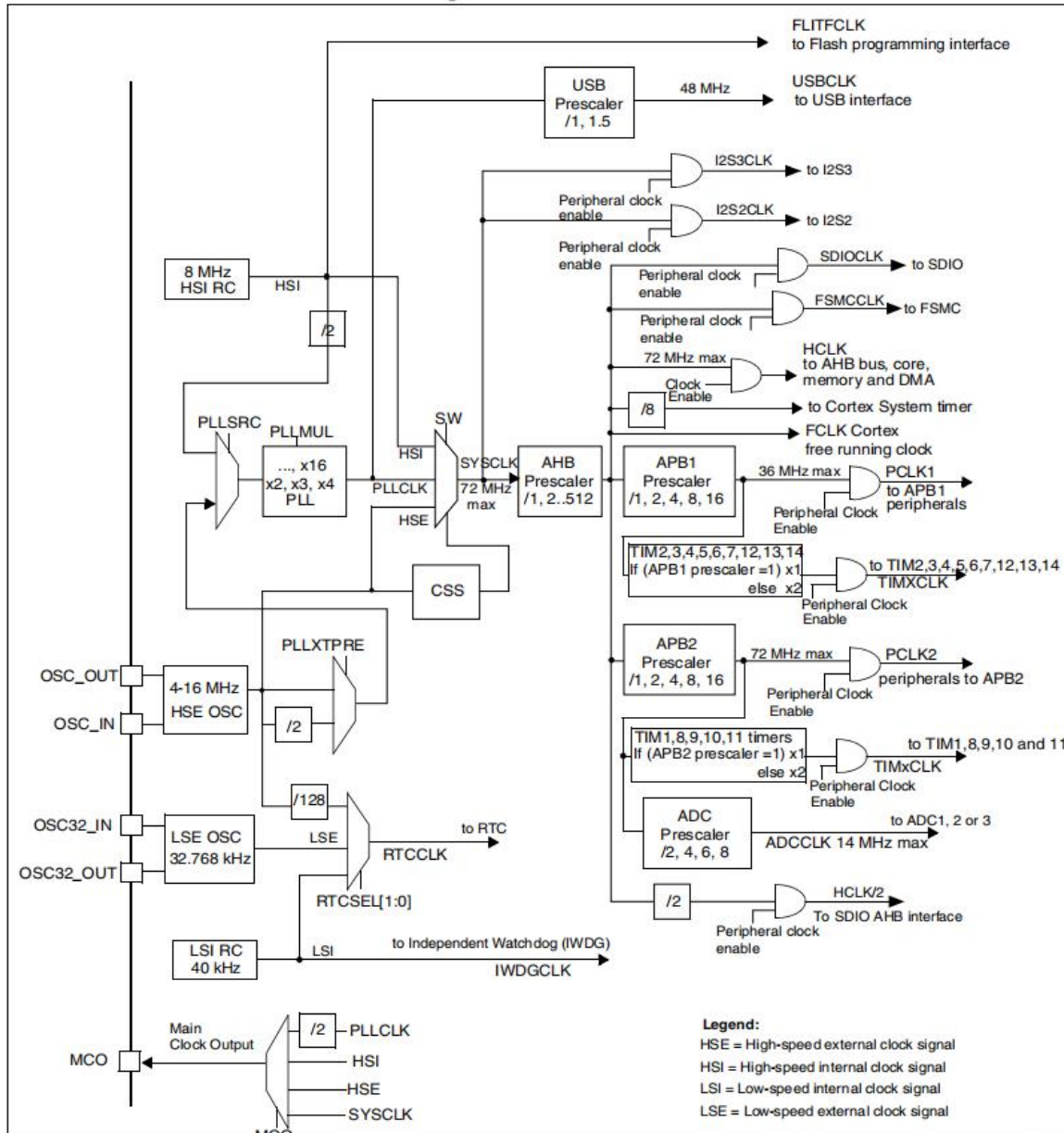
The devices have the following two secondary clock sources:

- 40 kHz low speed internal RC (LSI RC), which drives the independent watchdog and optionally the RTC used for Auto-wakeup from Stop/Standby mode.
- 32.768 kHz low speed external crystal (LSE crystal), which optionally drives the real-time clock (RTCCLK)

Each clock source can be switched on or off independently when it is not used, to optimize power consumption.

STM32F103 FAMILY CLOCK DIAGRAM

Figure 8. Clock tree



MX_GPIO_Init

```
14 ⊖ /**
15  * @brief GPIO Initialization Function
16  * @param None
17  * @retval None
18  */
19 ⊖ void MX_GPIO_Init(void)
20 {
21  // GPIO_InitTypeDef GPIO_InitStruct = {0};
22
23  /* GPIO Ports Clock Enable */
24  __HAL_RCC_GPIOC_CLK_ENABLE();
25  __HAL_RCC_GPIOD_CLK_ENABLE();
26  __HAL_RCC_GPIOA_CLK_ENABLE();
27  __HAL_RCC_GPIOB_CLK_ENABLE();
28  |
29 ⊖ // /*Configure GPIO pin Output Level */
30 // HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);
31 //
32 LED_Init();
33 GPIO_Relay_Init();
34 Buzzer_Init();
35 SPI_CS_Init();
36 ZeroPoint_Detection_Pin_Init();
37 }
38
```

Input Output Pins Initialization

```
38
39 void LED_Init(void){
40
41     GPIO_InitTypeDef GPIO_InitStructure = {0};
42
43     GPIO_InitStructure.Mode = GPIO_MODE_OUTPUT_PP;
44     GPIO_InitStructure.Pull = GPIO_PULLUP;
45     GPIO_InitStructure.Speed = GPIO_SPEED_FREQ_HIGH;
46     GPIO_InitStructure.Pin = LED2_PIN;
47     HAL_GPIO_Init(LED2_GPIO_PORT, &GPIO_InitStructure);
48
49 }
50
```

```
102
103 void ZeroPoint_Detection_Pin_Init(void){
104     GPIO_InitTypeDef GPIO_InitStructure = {0};
105
106     GPIO_InitStructure.Pin = ZERO_POINT_DETECTION_PIN;
107     GPIO_InitStructure.Mode = GPIO_MODE_IT_RISING;
108     GPIO_InitStructure.Pull = GPIO_NOPULL;
109     HAL_GPIO_Init(ZERO_POINT_DETECTION_PORT, &GPIO_InitStructure);
110
111     #if(VERSION_EBOX == 2)
112         /* EXTI interrupt init*/
113         HAL_NVIC_SetPriority(EXTI9_5_IRQn, 0, 0);
114         HAL_NVIC_EnableIRQ(EXTI9_5_IRQn);
115     #else
116         HAL_NVIC_SetPriority(EXTI15_10_IRQn, 0, 0);
117         HAL_NVIC_EnableIRQ(EXTI15_10_IRQn);
118     #endif
119 }
120
```

```
--
54 void GPIO_Relay_Init(void){
55     GPIO_InitTypeDef GPIO_InitStructure = {0};
56
57     GPIO_InitStructure.Mode = GPIO_MODE_OUTPUT_PP;
58     GPIO_InitStructure.Pull = GPIO_PULLUP;
59     GPIO_InitStructure.Speed = GPIO_SPEED_FREQ_HIGH;
60
61     GPIO_InitStructure.Pin = RELAY_PIN_0;
62     HAL_GPIO_Init(RELAY_PORT_0, &GPIO_InitStructure);
63
64     GPIO_InitStructure.Pin = RELAY_PIN_1;
65     HAL_GPIO_Init(RELAY_PORT_1, &GPIO_InitStructure);
66     GPIO_InitStructure.Pin = RELAY_PIN_2;
67     HAL_GPIO_Init(RELAY_PORT_2, &GPIO_InitStructure);
68     GPIO_InitStructure.Pin = RELAY_PIN_3;
69     HAL_GPIO_Init(RELAY_PORT_3, &GPIO_InitStructure);
70     GPIO_InitStructure.Pin = RELAY_PIN_4;
71     HAL_GPIO_Init(RELAY_PORT_4, &GPIO_InitStructure);
72     GPIO_InitStructure.Pin = RELAY_PIN_5;
73     HAL_GPIO_Init(RELAY_PORT_5, &GPIO_InitStructure);
74     GPIO_InitStructure.Pin = RELAY_PIN_6;
75     HAL_GPIO_Init(RELAY_PORT_6, &GPIO_InitStructure);
76     GPIO_InitStructure.Pin = RELAY_PIN_7;
77     HAL_GPIO_Init(RELAY_PORT_7, &GPIO_InitStructure);
78     GPIO_InitStructure.Pin = RELAY_PIN_8;
79     HAL_GPIO_Init(RELAY_PORT_8, &GPIO_InitStructure);
80     GPIO_InitStructure.Pin = RELAY_PIN_9;
81     HAL_GPIO_Init(RELAY_PORT_9, &GPIO_InitStructure);
82     #if(VERSION_EBOX == 2)
83         GPIO_InitStructure.Pin = PD2_RELAY_ENABLE_PIN;
84         HAL_GPIO_Init(PD2_RELAY_ENABLE_PORT, &GPIO_InitStructure);
85     #endif
86 }
```

UART3_Init

```
78 /**
79  * @brief USART3 Initialization Function
80  * @param None
81  * @retval None
82  */
83 void UART3_Init(void)
84 {
85     Uart3Handle.Instance = USART3;
86     Uart3Handle.Init.BaudRate = 115200;
87     Uart3Handle.Init.WordLength = UART_WORDLENGTH_8B;
88     Uart3Handle.Init.StopBits = UART_STOPBITS_1;
89     Uart3Handle.Init.Parity = UART_PARITY_NONE;
90     Uart3Handle.Init.Mode = UART_MODE_TX_RX;
91     Uart3Handle.Init.HwFlowCtl = UART_HWCONTROL_NONE;
92     Uart3Handle.Init.OverSampling = UART_OVERSAMPLING_16;
93     if (HAL_UART_Init(&Uart3Handle) != HAL_OK) {
94         Error_Handler();
95     }
96 }
```

Timer_Init

Configure the TIM peripheral

In this example TIM3 input clock (TIM3CLK) is set to APB1 clock (PCLK1) x2, since APB1 prescaler is set to 4 (0x100).

$TIM3CLK = PCLK1 * 2$

- $PCLK1 = HCLK / 2$
- $\Rightarrow TIM3CLK = PCLK1 * 2 = (HCLK / 2) * 2 = HCLK = SystemCoreClock$
- To get TIM3 counter clock at 10 KHz, the Prescaler is computed as following:
- $Prescaler = (TIM3CLK / TIM3\ counter\ clock) - 1$
- $Prescaler = (SystemCoreClock / 10\ KHz) - 1$

Note:

- SystemCoreClock variable holds HCLK frequency and is defined in **system_stm32f1xx.c** file.
- Each time the core clock (HCLK) changes, user had to update SystemCoreClock variable value. Otherwise, any configuration based on this variable will be incorrect.
- This variable is updated in three ways:
 - 1) by calling CMSIS function SystemCoreClockUpdate()
 - 2) by calling HAL API function HAL_RCC_GetSysClockFreq()
 - 3) each time HAL_RCC_ClockConfig() is called to configure the system clock frequency



Timer_Init

```
37
38  /* Compute the prescaler value to have TIMx counter clock equal to 10000 Hz */
39  uwPrescalerValue = (uint32_t)(SystemCoreClock / 10000) - 1;
40
41  /* Set TIMx instance */
42  TimHandle.Instance = TIMx;
43
44  /* Initialize TIMx peripheral as follows:
45   + Period = 10000 - 1
46   + Prescaler = (SystemCoreClock/10000) - 1
47   + ClockDivision = 0
48   + Counter direction = Up
49  */
50  TimHandle.Init.Period          = 100 - 1;
51  TimHandle.Init.Prescaler       = uwPrescalerValue;
52  TimHandle.Init.ClockDivision   = 0;
53  TimHandle.Init.CounterMode     = TIM_COUNTERMODE_UP;
54  TimHandle.Init.RepetitionCounter = 0;
55  TimHandle.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
56
57  if (HAL_TIM_Base_Init(&TimHandle) != HAL_OK)
58  {
59      /* Initialization Error */
60      Error_Handler();
61  }
62
63  /*##-2- Start the TIM Base generation in interrupt mode #####*/
64  /* Start Channel1 */
65  if (HAL_TIM_Base_Start_IT(&TimHandle) != HAL_OK)
66  {
67      /* Starting Error */
68      Error_Handler();
69  }
70
71  // HAL_NVIC_SetPriority((IRQn_Type)(TIM3_IRQn), 0x0, 0);
72  }
73
```

An FSM function

```
enum {St_Intro, St_Voltmeter, St_Temperature, St_Clock,} State_Machine;  
unsigned char State = St_Intro;  
void FSM(){  
    switch (State)  
    {  
        case St_Intro:    Intro();  
        break;  
        case St_Voltmeter:    Voltmeter();  
        break;  
        case St_Temperature: Temperature();  
        break;  
        case St_Clock: Clock();  
        break;  
    }  
}
```

