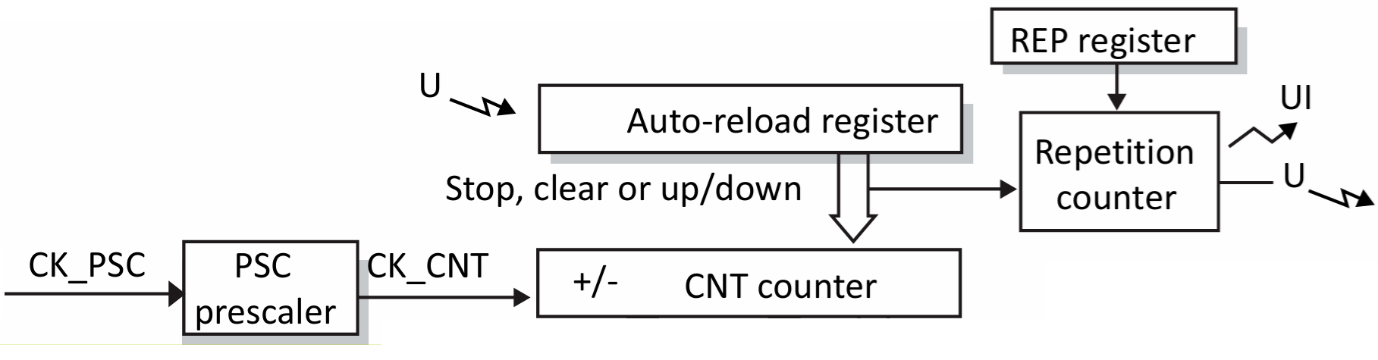
|  |  |
| --- | --- |
| TRƯỜNG ĐH SƯ PHẠM KỸ THUẬT TPHCM  **KHOA CƠ KHÍ CHẾ TẠO MÁY**  **BỘ MÔN CƠ ĐIỆN TỬ**  **Họ và tên:** Võ Hồng Quân  **Mã số SV:** 22134012 | **BÁO CÁO**  **THỰC TẬP LẬP TRÌNH VI XỬ LÝ**  Ngày:  Tuần học: **05**  Nội dung: **Timer, ngắt timer và ứng dụng** |

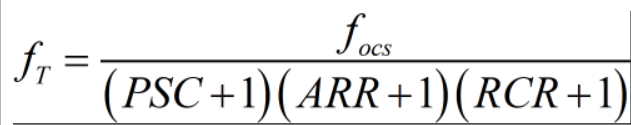
1. **Khảo sát các bộ timer trong STM32:**
   1. **Ý nghĩa các thanh ghi Timer cơ bản:**

****



|  |  |  |
| --- | --- | --- |
| STT | Thanh ghi | Chức năng |
| 1 | TIMx\_PSC | Chia tần số xung clock đầu vào của bộ định thời. Giá trị này giúp giảm tốc độ đếm của Timer |
| 2 | TIMx\_ARR | Giá trị đếm tối đa của Timer. Khi bộ đếm đạt giá trị này, một sự kiện "Update" xảy ra và bộ đếm có thể được đặt lại về 0 hoặc giá trị khác |
| 3 | TIMx\_RCR | Xác định số lần lặp lại của sự kiện Update trước khi ngắt được kích hoạt |
| 4 | TIMx\_CNT | Chứa giá trị hiện tại của bộ đếm. Giá trị này tăng hoặc giảm tùy thuộc vào chế độ đếm |

* 1. **Công thức tính thời gian tràn timer:**

****

1. **Thư viện HAL sử dụng timer:**

|  |  |  |
| --- | --- | --- |
| STT | Lệnh | Giải thích lệnh |
| 1 | HAL\_TIM\_Base\_Start  (TIM\_HandleTypeDef \*htim) | Bắt đầu Timer ở chế độ cơ bản (không sử dụng ngắt). Timer sẽ chạy liên tục từ CNT = 0 đến ARR rồi reset về 0. |
| 2 | HAL\_TIM\_Base\_Stop  (TIM\_HandleTypeDef \*htim) | Dừng Timer đang chạy. Giá trị bộ đếm CNT sẽ giữ nguyên tại thời điểm dừng. |
| 3 | HAL\_TIM\_Base\_Start\_IT  (TIM\_HandleTypeDef \*htim) | Bắt đầu Timer ở chế độ ngắt (Interrupt Mode). Khi Timer đạt giá trị ARR, nó sẽ kích hoạt ngắt và gọi HAL\_TIM\_PeriodElapsedCallback(). |
| 4 | HAL\_TIM\_Base\_Stop\_IT  (TIM\_HandleTypeDef \*htim) | Dừng Timer ở chế độ ngắt. Bộ đếm CNT ngừng lại và ngắt Timer sẽ bị vô hiệu hóa. |
| 5 | HAL\_TIM\_Encoder\_Start  (TIM\_HandleTypeDef \*htim,  uint32\_t Channel); | Kích hoạt Timer ở chế độ Encoder Mode để đọc giá trị từ encoder quay. |
| 6 | HAL\_TIM\_Encoder\_Stop  (TIM\_HandleTypeDef \*htim,  uint32\_t Channel) | Dừng Timer ở chế độ Encoder Mode khi không cần đọc tín hiệu từ encoder nữa. |
| 7 | \_\_HAL\_TIM\_GET\_FLAG  (\_\_HANDLE\_\_, \_\_FLAG\_\_) | Kiểm tra trạng thái cờ của Timer (ví dụ: TIM\_FLAG\_UPDATE, TIM\_FLAG\_CC1). Nếu cờ được set, có thể xác nhận sự kiện đã xảy ra. |
| 8 | \_\_HAL\_TIM\_CLEAR\_FLAG  (\_\_HANDLE\_\_, \_\_FLAG\_\_) | Xóa cờ ngắt của Timer, thường được sử dụng sau khi xử lý sự kiện để tránh lặp lại không mong muốn. |
| 9 | \_\_HAL\_TIM\_SET\_COUNTER  (\_\_HANDLE\_\_, \_\_COUNTER\_\_) | Đặt giá trị của bộ đếm Timer (CNT). Dùng khi cần reset Timer hoặc thay đổi giá trị đếm. |
| 10 | \_\_HAL\_TIM\_GET\_COUNTER  (\_\_HANDLE\_\_) | Đọc giá trị hiện tại của bộ đếm Timer. Hữu ích để kiểm tra thời gian đã trôi qua hoặc xác định vị trí encoder. |

1. **Lập trình**
   1. **Sử dụng Timer tạo trì hoãn**

Tính toán các giá trị Timer Base để tạo trì hoãn 1 giây. Viết chương trình sử dụng hàm trì hoãn này để sáng tắt đèn D1 theo chế độ hỏi vòng (polling)

Bài làm:

* Hàm main

|  |
| --- |
| int main(void)  {  HAL\_Init();  SystemClock\_Config();  MX\_GPIO\_Init();  MX\_TIM1\_Init();  HAL\_GPIO\_WritePin(D1\_GPIO\_Port, D1\_Pin, GPIO\_PIN\_RESET);  HAL\_TIM\_Base\_Start(&htim1); // TIM1 start counting  while (1)  {  \_\_HAL\_TIM\_CLEAR\_FLAG(&htim1, TIM\_FLAG\_UPDATE); // Update flag = 0  \_\_HAL\_TIM\_SET\_COUNTER(&htim1, 0); // CNT = 0)  while (\_\_HAL\_TIM\_GET\_FLAG(&htim1, TIM\_FLAG\_UPDATE)==0); // Waiting  HAL\_GPIO\_TogglePin(D1\_GPIO\_Port, D1\_Pin);  }  } |

* Hàm TIM1

|  |
| --- |
| static void MX\_TIM1\_Init(void)  {  /\* USER CODE BEGIN TIM1\_Init 0 \*/  /\* USER CODE END TIM1\_Init 0 \*/  TIM\_ClockConfigTypeDef sClockSourceConfig = {0};  TIM\_MasterConfigTypeDef sMasterConfig = {0};  /\* USER CODE BEGIN TIM1\_Init 1 \*/  /\* USER CODE END TIM1\_Init 1 \*/  htim1.Instance = TIM1;  htim1.Init.Prescaler = 999;  htim1.Init.CounterMode = TIM\_COUNTERMODE\_UP;  htim1.Init.Period = 15999;  htim1.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1;  htim1.Init.RepetitionCounter = 0;  htim1.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_DISABLE;  if (HAL\_TIM\_Base\_Init(&htim1) != HAL\_OK)  {  Error\_Handler();  }  sClockSourceConfig.ClockSource = TIM\_CLOCKSOURCE\_INTERNAL;  if (HAL\_TIM\_ConfigClockSource(&htim1, &sClockSourceConfig) != HAL\_OK)  {  Error\_Handler();  }  sMasterConfig.MasterOutputTrigger = TIM\_TRGO\_RESET;  sMasterConfig.MasterSlaveMode = TIM\_MASTERSLAVEMODE\_DISABLE;  if (HAL\_TIMEx\_MasterConfigSynchronization(&htim1, &sMasterConfig) != HAL\_OK)  {  Error\_Handler();  }  } |

* 1. **Sử dụng Timer đếm thời gian tác động tại GPIO**

Viết chương trình đếm thời gian nhấn nút (P1) và hiển thị lên LCD.

Bài làm:

|  |
| --- |
| uint16\_t press\_time = 0;  void DisplayTime() {  char timeString[16];  sprintf(timeString, "Time Press: %u", press\_time);  LCD\_GotoXY(0, 0);  LCD\_PutString((uint8\_t \*)timeString);  }  int main(void)  {  HAL\_Init();  SystemClock\_Config();  MX\_GPIO\_Init();  LCD\_Init();  MX\_TIM1\_Init();  HAL\_TIM\_Base\_Start(&htim1); // TIM1 b?t d?u ch?y    Light\_cmd(D1, D\_OFF);  Light\_cmd(D2, D\_OFF);  while (1)  {  P1\_STATUS = HAL\_GPIO\_ReadPin(P1\_GPIO\_Port, P1\_Pin);    if ( P1\_STATUS == P\_ACTIVE ) {  HAL\_Delay(1);  \_\_HAL\_TIM\_SET\_COUNTER(&htim1, 0);  uint32\_t start\_tick = HAL\_GetTick();  while (HAL\_GPIO\_ReadPin(P1\_GPIO\_Port, P1\_Pin) == GPIO\_PIN\_RESET) {  if (HAL\_GetTick() - start\_tick > 20000) break;  }  press\_time = \_\_HAL\_TIM\_GET\_COUNTER(&htim1);  DisplayTime();  }  press\_time = 0;  }  } |

* 1. **Sử dụng ngắt timer tạo vòng quét LED 7 đoạn.**

Viết chương trình tạo ngắt timer 1ms; trong chương trình xử lý ngắt timer thực hiện quét LED 7 đoạn hiển thị theo giá trị biến n.

Bài làm:

|  |
| --- |
| // Timer Interrupt Callback for LED Scanning  void HAL\_TIM\_PeriodElapsedCallback(TIM\_HandleTypeDef \*htim) {  if (htim->Instance == TIM1) { // Xác nh?n là TIM1  \_\_HAL\_TIM\_CLEAR\_FLAG(htim, TIM\_FLAG\_UPDATE); // Xóa c? Update (thu?ng t? d?ng)  \_\_HAL\_TIM\_SET\_COUNTER(htim, 0); // Reset CNT d? d?m b?o th?i gian ?n d?nh  // B?ng ánh x? v? trí (theo gi? d?nh LED 7 do?n du?c chia 5 ph?n)  uint8\_t pos[5] = {0x08, 0x10, 0x20, 0x40, 0x80};  uint8\_t s[] = "34012"; // Chu?i c?n hi?n th?  static uint8\_t current\_digit = 0; // V? trí hi?n t?i (0-4)  uint8\_t code;  if (s[current\_digit] == '\0') {  code = LED\_OFF;  } else {  code = LED\_decode\_char(s[current\_digit]);  }  // Ghép d? li?u: byte cao là m?t n? v? trí, byte th?p là mã 7 do?n  uint16\_t d16 = ((uint16\_t)pos[current\_digit] << 8) | code;  Data16\_put(d16);  // Chuy?n sang ch? s? ti?p theo  current\_digit++;  if (current\_digit >= 5) {  current\_digit = 0;  }  }  } |

|  |
| --- |
| static void MX\_TIM1\_Init(void)  {  TIM\_ClockConfigTypeDef sClockSourceConfig = {0};  TIM\_MasterConfigTypeDef sMasterConfig = {0};  htim1.Instance = TIM1;  htim1.Init.Prescaler = 15; // 16 MHz/16 = 1 MHz (1 tick = 1 µs)  htim1.Init.CounterMode = TIM\_COUNTERMODE\_UP;  htim1.Init.Period = 999; // 1000 ticks = 1ms  htim1.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1;  htim1.Init.RepetitionCounter = 0;  htim1.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_DISABLE;  if (HAL\_TIM\_Base\_Init(&htim1) != HAL\_OK)  {  Error\_Handler();  }  sClockSourceConfig.ClockSource = TIM\_CLOCKSOURCE\_INTERNAL;  if (HAL\_TIM\_ConfigClockSource(&htim1, &sClockSourceConfig) != HAL\_OK)  {  Error\_Handler();  }  sMasterConfig.MasterOutputTrigger = TIM\_TRGO\_RESET;  sMasterConfig.MasterSlaveMode = TIM\_MASTERSLAVEMODE\_DISABLE;  if (HAL\_TIMEx\_MasterConfigSynchronization(&htim1, &sMasterConfig) != HAL\_OK)  {  Error\_Handler();  }  } |

* 1. **Lập trình ứng dụng 1:** Tạo trì hoãn tốc độ BAUD và viết hàm softuart.

Viết chương trình tạo hàm trì hoãn **BAUD\_delay**(uint32 ***baud***) để tạo trì hoãn truyền dữ liệu theo tham số ***baud***. Ví dụ: BAUD\_delay(9600) sẽ tạo delay 1/9600 (giây).

Sử dụng hàm **BAUD\_delay** đã viết ở trên để viết hàm truyền dữ liệu **SoftUART\_putc**(uint8\_t **ch**) và **SoftUART\_print**(uint8\_t \* **s**).

Bài làm:

|  |
| --- |
| #define SOFTUART\_BAUD 9600  #define SOFTUART\_TX\_GPIO\_Port GPIOA  #define SOFTUART\_TX\_Pin GPIO\_PIN\_9  // BAUD\_delay, SoftUART\_putc, SoftUART\_print nhu dã d?nh nghia ? trên  void BAUD\_delay(uint32\_t baud) {  uint32\_t delay\_us = 1000000 / baud;  \_\_HAL\_TIM\_SET\_COUNTER(&htim1, 0);  while(\_\_HAL\_TIM\_GET\_COUNTER(&htim1) < delay\_us);  }  void SoftUART\_putc(uint8\_t ch) {  HAL\_GPIO\_WritePin(SOFTUART\_TX\_GPIO\_Port, SOFTUART\_TX\_Pin, GPIO\_PIN\_RESET); // Start bit  BAUD\_delay(SOFTUART\_BAUD);    for (int i = 0; i < 8; i++) {  if (ch & (1 << i))  HAL\_GPIO\_WritePin(SOFTUART\_TX\_GPIO\_Port, SOFTUART\_TX\_Pin, GPIO\_PIN\_SET);  else  HAL\_GPIO\_WritePin(SOFTUART\_TX\_GPIO\_Port, SOFTUART\_TX\_Pin, GPIO\_PIN\_RESET);  BAUD\_delay(SOFTUART\_BAUD);  }    HAL\_GPIO\_WritePin(SOFTUART\_TX\_GPIO\_Port, SOFTUART\_TX\_Pin, GPIO\_PIN\_SET); // Stop bit  BAUD\_delay(SOFTUART\_BAUD);  }  void SoftUART\_print(uint8\_t \*s) {  while (\*s) {  SoftUART\_putc(\*s++);  }  }  int main(void) {  HAL\_Init();  SystemClock\_Config();  MX\_GPIO\_Init();  LCD\_Init();  MX\_TIM1\_Init();  // B?t d?u TIM1 (TIM1 c?n du?c c?u hình sao cho m?i tick = 1µs)  HAL\_TIM\_Base\_Start(&htim1);  while (1) {  SoftUART\_print((uint8\_t \*)"Hello, SoftUART!\r\n");  HAL\_Delay(1000); // G?i chu?i m?i 1 giây  }  } |

* 1. **Lập trình ứng dụng 2:** Viết thư viện giao tiếp cảm biến siêu âm SRF04.

Bài làm:

* 1. **Lập trình ứng dụng 3:** Viết chương trình đo tốc độ quay của encoder hiển thị lên LED 7 đoạn.

Bài làm: