

Busan Software Meister High School

# ***MICROPROCESSOR***

2309 양유빈

20231012 마이크로프로세서

# SysTick 타이머 및 SysTick 관련 HAL 라이브러리

SysTick-related HAL libraries

- 24비트 다운 카운터 기능을 갖는 시스템 타이머
- 특정 값 설정 시 클록에 의해 1씩 감소하다가 타이머 값 0 되는 순간 SysTick 인터럽트를 발생, 동시에 다시 정해진 값이 로드되어 다운카운트 반복
- STM32CubeIDE에서 초기화 프로그램을 생성할 때, 1ms 마다 SysTick 타이머 인터럽트가 발생하도록 자동으로 생성됨
- SysTick 핸들러 : 1ms 마다 SysTick 인터럽트 발생하면 SysTick\_Handler( )가 실행됨

형태	__weak void HAL_IncTick(void)
설명	uwTick 변수를 1(=uwTickFreq) 증가 시킴
파라미터	없음
리턴값	없음
예시	시스템에서 기준 시간을 1ms 증가시키고자 할 때의 코딩: HAL_IncTick( );

# SysTick 관련 HAL 라이브러리

SysTick-related HAL libraries

형태	__weak uint32_t HAL_GetTick(void)
설명	현재의 uwTick 변수값을 돌려줌
파라미터	없음
리턴값	현재의 uwTick 값
예시	시스템 초기 시작부터 지금까지 지나간 시간(ms 단위)을 알고 싶을 때의 코딩: time = HAL_GetTick( );

# HELP 오른쪽

## FND STM32CubeIDE Settings

```

56  /* USER CODE BEGIN 0 */
57  typedef struct fnd {
58      GPIO_TypeDef *port;
59      uint16_t pin;
60  } FND;
61  FND value[8] = {
62      {GPIOB, GPIO_PIN_0}, {GPIOB, GPIO_PIN_1},
63      {GPIOB, GPIO_PIN_2}, {GPIOB, GPIO_PIN_3},
64      {GPIOB, GPIO_PIN_4}, {GPIOB, GPIO_PIN_5},
65      {GPIOB, GPIO_PIN_6}, {GPIOB, GPIO_PIN_7}
66  };
67  FND sel[4] = {
68      {GPIOA, GPIO_PIN_10}, {GPIOA, GPIO_PIN_11},
69      {GPIOC, GPIO_PIN_4}, {GPIOC, GPIO_PIN_3}
70  };
71
72  /* USER CODE BEGIN PV */
73  uint8_t help[] = {0x76, 0x79, 0x38, 0x73};
74  /* USER CODE END PV */
75
76
77
78
79
80
81
82
83
84
85
86

```

```
71 void display_fnd(uint8_t data, uint8_t position, uint32_t time) {
72     int i;
73     for (i = 0; i < 4; i++) {
74         if(i == position)
75             HAL_GPIO_WritePin(sel[i].port, sel[i].pin, 1);
76         else
77             HAL_GPIO_WritePin(sel[i].port, sel[i].pin, 0);
78     }
79     for(i = 0; i < 8; i++) {
80         if((data & (1 << i)) != 0)
81             HAL_GPIO_WritePin(value[i].port, value[i].pin, 1);
82         else
83             HAL_GPIO_WritePin(value[i].port, value[i].pin, 0);
84     }
85     HAL_Delay(time);
86 }
```

# HELP 오른쪽

FND STM32CubeIDE Settings

```
/* infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    for(int i = 4; i > 0; i--) {
        for(int j = 0; j < 50; j++) {
            display_fnd(help[0], (3+i)%4, 5);
            display_fnd(help[1], (2+i)%4, 5);
            display_fnd(help[2], (1+i)%4, 5);
            display_fnd(help[3], (0+i)%4, 5);
        }
    }
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
}
```

# 왼쪽

FND 7 display code

```
123      /* USER CODE BEGIN WHILE */
124      while (1)
125      {
126      #if 0
127          for(int i = 4; i > 0; i--) {
128              for(int j = 0; j < 50; j++) {
129                  display_fnd(help[0], (3+i)%4, 5);
130                  display_fnd(help[1], (2+i)%4, 5);
131                  display_fnd(help[2], (1+i)%4, 5);
132                  display_fnd(help[3], (0+i)%4, 5);
133              }
134          }
135      #endif
136          for(int i = 0; i < 4; i++) {
137              for(int j = 0; j < 50; j++) {
138                  display_fnd(help[0], (3+i)%4, 5);
139                  display_fnd(help[1], (2+i)%4, 5);
140                  display_fnd(help[2], (1+i)%4, 5);
141                  display_fnd(help[3], (0+i)%4, 5);
142              }
143          }
144      /* USER CODE END WHILE */
145
146      /* USER CODE BEGIN 3 */
147      }
148      /* USER CODE END 3 */
---
```

# 광센서의 ADC값 출력

ADC value output of optical sensor

```
22 /* Private Includes -----
23 /* USER CODE BEGIN Includes */
24 #include <stdio.h>
25 /* USER CODE END Includes */
41
42 /* Private variables -----
43 ADC_HandleTypeDef hadc1;
44
45 TIM_HandleTypeDef htim2;
46
47 UART_HandleTypeDef huart2;
48
49 /* USER CODE BEGIN PV */
50 uint8_t number[] = {
51     0x3f,
52     0x06,
53     0x5b,
54     0x4f,
55     0x66,
56     0x6d,
57     0x7d,
58     0x07,
59     0x7f,
60     0x67
61 };
62 int adc_value;
63 volatile int ADC_1, ADC_2, ADC_3, ADC_4;
64 /* USER CODE END PV */
```

```
77 /* USER CODE BEGIN 0 */
78 int __io_putchar(int ch) {
79     HAL_UART_Transmit(&huart2, (uint8_t *)&ch, 1, 100);
80     if(ch == '\n'){
81         HAL_UART_Transmit(&huart2, (uint8_t *)'\r', 1, 100);
82     }
83     return ch;
84 }
85 typedef struct FND{
86     GPIO_TypeDef *port;
87     uint16_t *pin;
88 }FND;
89
90 FND value[8] = {
91     {GPIOB, GPIO_PIN_0},
92     {GPIOB, GPIO_PIN_1},
93     {GPIOB, GPIO_PIN_2},
94     {GPIOB, GPIO_PIN_3},
95     {GPIOB, GPIO_PIN_4},
96     {GPIOB, GPIO_PIN_5},
97     {GPIOB, GPIO_PIN_6},
98     {GPIOB, GPIO_PIN_7}
99 };
101 FND sel[4] = {
102     {GPIOA, GPIO_PIN_10},
103     {GPIOA, GPIO_PIN_11},
104     {GPIOC, GPIO_PIN_4},
105     {GPIOC, GPIO_PIN_3}
106 };
107
```

# 광센서의 ADC값 출력

ADC value output of optical sensor

```
108 void display(uint8_t num, uint8_t position, uint32_t time){
109     int i;
110     for(i = 0; i < 4; i++){
111         if(i == position){
112             HAL_GPIO_WritePin(sel[i].port, sel[i].pin, 1);
113         }
114         else{
115             HAL_GPIO_WritePin(sel[i].port, sel[i].pin, 0);
116         }
117     }
118     for(i = 0; i < 8; i++){
119         if((num & (1<<i)) != 0){
120             HAL_GPIO_WritePin(value[i].port, value[i].pin, 1);
121         }
122         else{
123             HAL_GPIO_WritePin(value[i].port, value[i].pin, 0);
124         }
125     }
126     HAL_Delay(time);
127 }
128 /* USER CODE END 0 */
```

```
156 /* Initialize all configured peripherals */
157 MX_GPIO_Init();
158 MX_ADC1_Init();
159 MX_TIM2_Init();
160 MX_USART2_UART_Init();
161 /* USER CODE BEGIN 2 */
162 HAL_ADCEx_Calibration_Start(&hadc1);
163 HAL_ADC_Start_IT(&hadc1);
164 HAL_TIM_Base_Start_IT(&htim2);
165 /* USER CODE END 2 */
166
167 /* Infinite loop */
168 /* USER CODE BEGIN WHILE */
169 while (1)
170 {
171     /* USER CODE END WHILE */
172     display(number[ADC_1], 3, 5);
173     display(number[ADC_2], 2, 5);
174     display(number[ADC_3], 1, 5);
175     display(number[ADC_4], 0, 5);
176     /* USER CODE BEGIN 3 */
177 }
178 /* USER CODE END 3 */
179 }
```

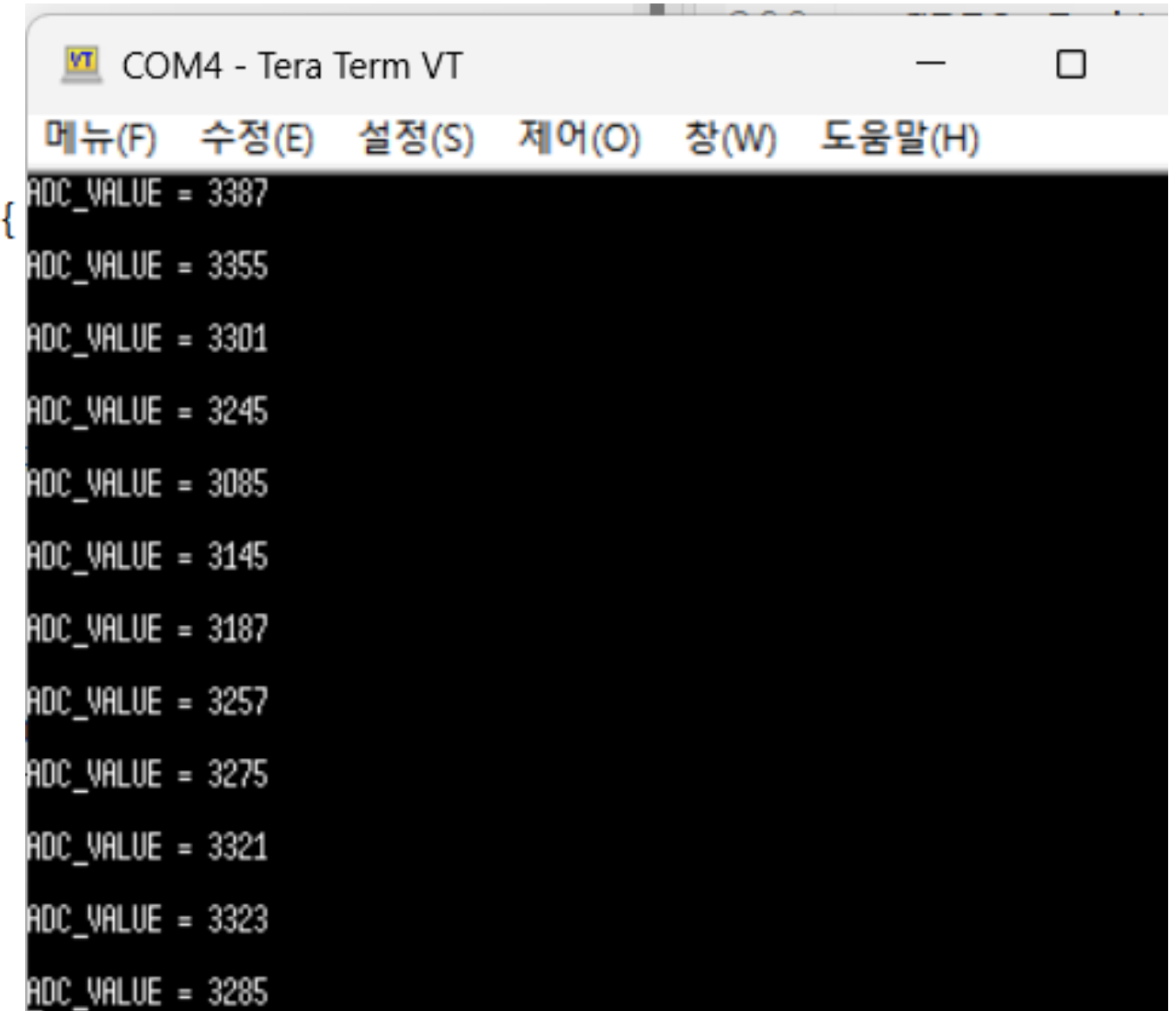


# 광센서의 ADC값 출력

ADC value output of optical sensor

```
/* USER CODE BEGIN 4 */
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) {
    if(htim==&htim2) {
        if (HAL_ADC_PollForConversion(&hadc1, 10) == HAL_OK) {
            adc_value = HAL_ADC_GetValue(&hadc1);
            ADC_1 = adc_value/1000;
            ADC_2 = (adc_value%1000)/100;
            ADC_3 = (adc_value%100)/10;
            ADC_4 = (adc_value%10)/1;
            printf("ADC_VALUE = %d\n\r", adc_value);
        }
    }
}
/* USER CODE END 4 */
```

// 결과

A screenshot of a Tera Term VT window titled "COM4 - Tera Term VT". The window has a menu bar with options: 메뉴(F), 수정(E), 설정(S), 제어(O), 창(W), and 도움말(H). The main area displays a list of ADC values, each preceded by "ADC\_VALUE =". The values are: 3387, 3355, 3301, 3245, 3085, 3145, 3187, 3257, 3275, 3321, 3323, and 3285.

```
COM4 - Tera Term VT
메뉴(F)  수정(E)  설정(S)  제어(O)  창(W)  도움말(H)
ADC_VALUE = 3387
ADC_VALUE = 3355
ADC_VALUE = 3301
ADC_VALUE = 3245
ADC_VALUE = 3085
ADC_VALUE = 3145
ADC_VALUE = 3187
ADC_VALUE = 3257
ADC_VALUE = 3275
ADC_VALUE = 3321
ADC_VALUE = 3323
ADC_VALUE = 3285
```