마이크로프로세서개론 프로그래밍과제 2

20181536 엄석훈

이번 과제를 위해 작성한 프로그램은 크게 초기화, EXTI핸들러, TIM핸들러 3개로 나누어서 구성되어 있다. 먼저 초기화에서는 L_SENSOR, R_SENSOR, EMERG 입력과 LED출력에 GPIO포트를 연결해주는 동작을 수행하며 아래 왼쪽과 같이 코드를 작성하였다. 먼저 GPIOC, GPIOA, AFIO를 사용함으로 RCC->APB2ENR |= 0x00000015; 로 bus를 연결하였다. LED출력은 PC0-PC7에 출력으로 연결되어 있음으로 GPIOC->CLR = 0x33333333; 으로 push-pull output으로 동작하도록 설정하였다. 그리고 PA8 - PA10을 사용하는 신호를 받는 입력으로 설정하였음으로 GPIOA->CRH = 0x00088800; 과 GPIOA->ODR = 0x00000000; 을 통해서 input pull-down으로 설정해주었다. 그리고 PA8, PA9, PA10이 눌려서 rising edge에서 인터럽트가 발생할 수 있도록 설정해주었다.

```
58  void init_port(void) {
59    RCC->APB2ENR |= 0x00000015;
60    GPIOC->CRL = 0x33333333;
61    GPIOA->CRH = 0x00088800;
62    GPIOA->ODR = 0x00000000;
63
64    EXTI->RTSR = 0x00001C00;
65    EXTI->IMR = 0x00001C00;
66    AFIO->EXTICR[2] = 0x00000000;
67    AFIO->EXTICR[3] = 0x00000000;
68    NVIC->ISER[1] |= 0x00000100;
69
70    return;
71  }
72
```

```
73  void init_timer(void){
74    RCC->APB1ENR |= 0x00000007;
75    RCC->APB2ENR |= 0x00000800;
76
77    TIM2->CR1 = 0x00000004;
78    TIM2->CR2 = 0x00000000;
79    TIM2->PSC = 0x00001C1F;
80    TIM2->ARR = 0x000003E7;
81    TIM2->DIER = 0x00000001;
82
83    TIM3->CR1 = 0x00000004;
84    TIM3->CR2 = 0x00000000;
85    TIM3->PSC = 0x00001C1F;
86    TIM3->ARR = 0x000003E7;
87    TIM3->DIER = 0x00000001;
88
89    TIM4->CR1 = 0x00000004;
90    TIM4->CR2 = 0x00000000;
91    TIM4->PSC = 0x00001C1F;
92    TIM4->ARR = 0x00004E1F;
93    TIM4->DIER = 0x00000001;
94
95    TIM1->CR1 = 0x00000004;
96    TIM1->CR2 = 0x00000000;
97    TIM1->PSC = 0x00001C1F;
98    TIM1->ARR = 0x00004E1F;
99    TIM1->DIER = 0x00000001;
100
101   NVIC->ISER[0] = 0x72000000;
102
103   return;
104 }
105
```

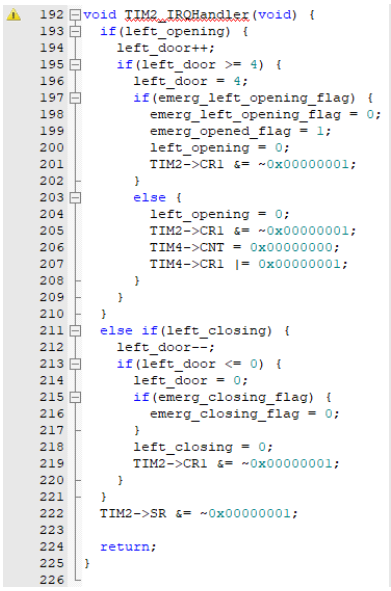
다음으로는 타이머를 초기화 해주었다. 이번 과제에서는 TIM1, TIM2, TIM3, TIM4까지 총 4개의 타이머를 사용한다. TIM2와 TIM3는 각각 왼쪽 문과 오른쪽 문이 열리고 닫힐 때 0.1초와 0.2초를 측정하는 타이머이고, TIM1과 TIM4는 각각 오른쪽 문과 왼쪽 문이 열린 시간 2초를 측정하는 타이머로 사용하였다. 이를 위해 위의 오른쪽 코드와 같이 작성하였다. 먼저 동일하게 모든 TIM를 bus에 연결해주고 모든 타이머를 동일하게 CR1 = 0x00000004, CR2 = 0x00000000, DIER = 0x00000001; 로 설정하여 overflow가 일어날 때만 update interrupt가 발생할 수 있도록 설정하였다. 그리고 PSC는 0x00001C1F 즉 7199로 고정하여 ARR값에 따라서 타이머의 update interrupt 주기가 결정되도록 초기화 하였다. 그리고 마짐가에는 NVIC->ISER[0] = 0x72000000; 을 통해서 TIM1, TIM2, TIM3, TIM4에서 인터럽트가 발생할 수 있도록 해주었다.

```
void EXTI15_10_IRQHandler(void) {
  if(EXTI->PR & 0x00000400) { //L_SEONSOR
    if((emerg_left_opening_flag || emerg_right_opening_flag || emerg_opened_flag || emerg_closing_flag) == 0) {
      left_opening_timer();
    }
    EXTI->PR = 0x00000400;
  }

  if(EXTI->PR & 0x00000800) { //R_SENSOR
    if((emerg_left_opening_flag || emerg_right_opening_flag || emerg_opened_flag || emerg_closing_flag) == 0) {
      right_opening_timer();
    }
    EXTI->PR = 0x00000800;
  }

  if(EXTI->PR & 0x00001000) { //EMERG
    if(emerg_opened_flag == 1) {
      emerg_left_opening_flag = 0;
      emerg_right_opening_flag = 0;
      emerg_opened_flag = 0;
      emerg_closing_flag = 1;
      left_closing_timer();
      right_closing_timer();
    }
    else {
      emerg_left_opening_flag = 1;
      emerg_right_opening_flag = 1;
      emerg_opened_flag = 0;
      emerg_closing_flag = 0;
      left_opening_timer();
      right_opening_timer();
      TIM4->CR1 &= ~0x00000001;
      TIM4->SR &= ~0x00000001;
      TIM1->CR1 &= ~0x00000001;
      TIM1->SR &= ~0x00000001;
    }
    EXTI->PR = 0x00001000;
  }

  return;
}
```

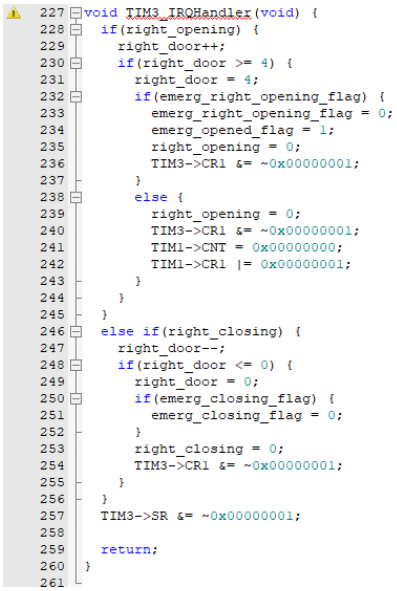
```
void left_opening_timer(void) {
  TIM2->CR1 &= ~0x00000001;
  left_opening = 1;
  left_closing = 0;
  TIM2->CNT = 0x00000000;
  TIM2->ARR = 0x000003E7; //0.1초
  TIM2->CR1 |= 0x00000001;
}

void left_closing_timer(void) {
  TIM2->CR1 &= ~0x00000001;
  left_opening = 0;
  left_closing = 1;
  TIM2->CNT = 0x00000000;
  TIM2->ARR = 0x000007CF; //0.2초
  TIM2->CR1 |= 0x00000001;
}

void right_opening_timer(void) {
  TIM3->CR1 &= ~0x00000001;
  right_opening = 1;
  right_closing = 0;
  TIM3->CNT = 0x00000000;
  TIM3->ARR = 0x000003E7;
  TIM3->CR1 |= 0x00000001;
}

void right_closing_timer(void){
  TIM3->CR1 &= ~0x00000001;
  right_opening = 0;
  right_closing = 1;
  TIM3->CNT = 0x00000000;
  TIM3->ARR = 0x000007CF;
  TIM3->CR1 |= 0x00000001;
}
```

다음으로 EXTI10_15핸들러는 좌측 위와 같이 작성하였다. 기본적으로 EXTI->PR에 마킹되어있는 bit를 확인해서 L_SENSOR와 R_SENSOR, EMERG중 어떤 신호가 들어왔는지 확인한다. 이때 신호가 동시에 들어올 수 있음으로 else if문이 아니라 if문 3개를 사용하였다. L_SENSOR와 R_SENSOR에서 신호가 들어오면 먼저 EMERG신호에 의해 문이 동작하고 있는지 여부를 확인하고 EMERG중이 아니라면 각 문을 여는 함수를 실행시킨다. 그리고 EMERG신호가 들어온 경우 문을 열어야 하는지 닫아야 하는지 확인해서 열어야 한다면 각종 flag를 업데이트하고 문을 여는 함수를 수행한다. 그리고 문을 닫아야 하는 경우에도 동일하게 각종 flag를 업데이트하고 문을 닫는 함수를 수행한다.

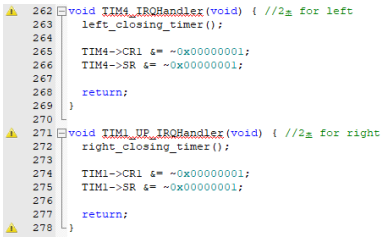
문을 열고 닫는데 사용하는 함수는 left, right따로 총 4개가 우측 위와 같이 작성되어 있다. 함수는 좌우가 대칭적인 형태로 문을 여는 opening_timer함수에서는 각각 TIM2와 TIM3를 리셋하고 flag를 업데이트하고 ARR값에 0x03E7 즉 999를 저장하여 update event 주기가 72MHz/((7199 + 1) * (999 + 1)) = 10Hz가 되도록 해서 0.1초 주기로 타이머에서 update interrupt가 발생하도록 만들고 타이머를 시작한다. 마찬가지로 문을 닫는 closing_timer함수도 TIM2와 tim3를 리셋하고 다시 flag를 업데이트하고 ARR값에 0x07CF즉 1999를 저장하여 update event 주기가 72MHz/((7199 + 1)*(1999 + 1)) = 5Hz즉 0.2초 주기로 타이머가 update interrupt가 발생하도록 만들어 주었다.



```c
void TIM2_IRQHandler(void) {
  if(left_opening) {
    left_door++;
    if(left_door >= 4) {
      left_door = 4;
      if(emerg_left_opening_flag) {
        emerg_left_opening_flag = 0;
        emerg_opened_flag = 1;
        left_opening = 0;
        TIM2->CR1 &= ~0x00000001;
      }
      else {
        left_opening = 0;
        TIM2->CR1 &= ~0x00000001;
        TIM4->CNT = 0x00000000;
        TIM4->CR1 |= 0x00000001;
      }
    }
  }
  else if(left_closing) {
    left_door--;
    if(left_door <= 0) {
      left_door = 0;
      if(emerg_closing_flag) {
        emerg_closing_flag = 0;
      }
      left_closing = 0;
      TIM2->CR1 &= ~0x00000001;
    }
  }
  TIM2->SR &= ~0x00000001;

  return;
}
```



```c
void TIM3_IRQHandler(void) {
  if(right_opening) {
    right_door++;
    if(right_door >= 4) {
      right_door = 4;
      if(emerg_right_opening_flag) {
        emerg_right_opening_flag = 0;
        emerg_opened_flag = 1;
        right_opening = 0;
        TIM3->CR1 &= ~0x00000001;
      }
      else {
        right_opening = 0;
        TIM3->CR1 &= ~0x00000001;
        TIM1->CNT = 0x00000000;
        TIM1->CR1 |= 0x00000001;
      }
    }
  }
  else if(right_closing) {
    right_door--;
    if(right_door <= 0) {
      right_door = 0;
      if(emerg_closing_flag) {
        emerg_closing_flag = 0;
      }
      right_closing = 0;
      TIM3->CR1 &= ~0x00000001;
    }
  }
  TIM3->SR &= ~0x00000001;

  return;
}
```



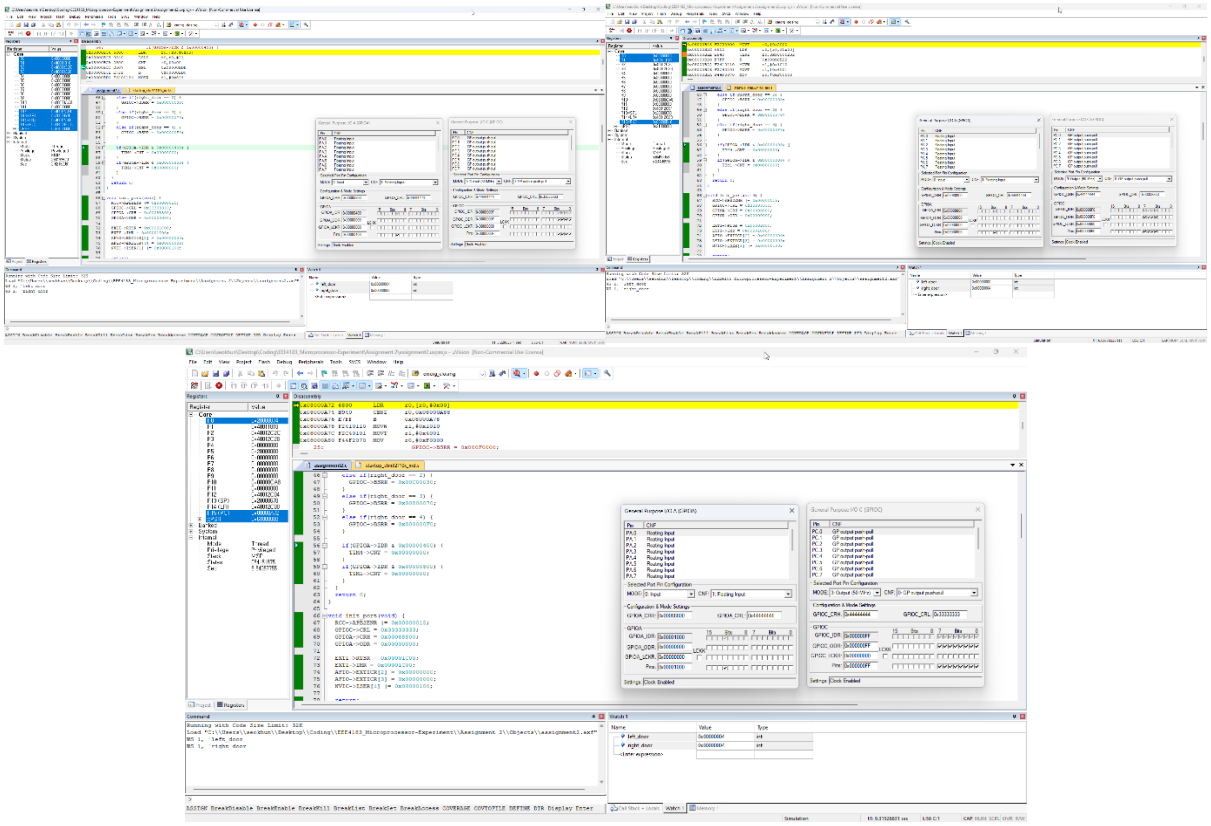
```c
void TIM4_IRQHandler(void) { //2초 for left
  left_closing_timer();

  TIM4->CR1 &= ~0x00000001;
  TIM4->SR &= ~0x00000001;

  return;
}

void TIM1_UP_IRQHandler(void) { //2초 for right
  right_closing_timer();

  TIM1->CR1 &= ~0x00000001;
  TIM1->SR &= ~0x00000001;

  return;
}
```

마지막으로 위와 같이 TIM2, TIM3, TIM4, TIM1_UP 인터럽트 핸들러를 작성해주었다. TIM2와 TIM3는 문이 열리고 닫힐 때 각 문의 0.1초, 0.2초를 측정하는 타이머이고 TIM4, TIM1_UP은 좌측과 우측 문이 2초동안 열렸는지 측정해주는 타이머이다. TIM2, TIM3 인터럽트 핸들러에서는 문을 여는 중이라면 문이 몇 칸 열렸는지 저장하는 left_door, right_door를 1씩 증가시킨다. 그리고 만약 4 이상이라면 EMERG신호가 눌려서 문이 열리는 중이었다면 관련 flag를 업데이트 하고 2초를 측정하는 타이머를 작동시키지 않는다. 만약 SENSOR에 의해 문이 열리는 중이었다면 2초를 측정하는 타이머를 작동시킨다. 마찬가지로 타이머 인터럽트에서 문이 닫히는 중이었다면 left_door, right_door를 1씩 감소시키고 만약 0이하라면 EMERG신호가 눌렸다면 EMERG상태를 종료하고 아니라면 해당 타이머를 끄고 초기상태로 돌아간다.

그리고 TIM4, TIM1_UP 인터럽트 핸들러에서는 update interrupt가 발생하였다는 것은 문이 열리고 2초가 지났다는 뜻임으로 closing_timer함수를 호출하여 문을 닫는 절차를 수행시키고 TIM4, TIM1은 필요없음으로 disable한다.

```
19  int main(void) {
20    init_port();
21    init_timer();
22
23    while(1) {
24      if(left_door == 0) {
25        GPIOC->BSRR = 0x000F0000;
26      }
27      else if(left_door == 1) {
28        GPIOC->BSRR = 0x00070008;
29      }
30      else if(left_door == 2) {
31        GPIOC->BSRR = 0x0003000C;
32      }
33      else if(left_door == 3) {
34        GPIOC->BSRR = 0x0001000E;
35      }
36      else if(left_door == 4) {
37        GPIOC->BSRR = 0x0000000F;
38      }
39
40      if(right_door == 0) {
41        GPIOC->BSRR = 0x00F00000;
42      }
43      else if(right_door == 1) {
44        GPIOC->BSRR = 0x00E00010;
45      }
46      else if(right_door == 2) {
47        GPIOC->BSRR = 0x00C00030;
48      }
49      else if(right_door == 3) {
50        GPIOC->BSRR = 0x00800070;
51      }
52      else if(right_door == 4) {
53        GPIOC->BSRR = 0x000000F0;
54      }
55
56      if(GPIOA->IDR & 0x00000400) {
57        TIM4->CNT = 0x00000000;
58      }
59      if(GPIOA->IDR & 0x00000800) {
60        TIM1->CNT = 0x00000000;
61      }
62    }
63    return 0;
64  }
65
```

마지막으로 이 프로그램의 main함수는 위와 같이 작성하였다. 간단하게 init_port()함수와 init_timer()함수를 호출하여 GPIO와 타이머를 초기화 해준다. 그리고 while문을 무한루프 돌면서 문이 열리고 닫히면서 바뀌는 left_door와 right_door에 따라서 GPIOC->BSRR 레지스터를 통해서 LED가 연결되어 있는 PC0 – PC7에서 출력되는 값을 바꿔준다. 그리고 추가적으로 센서에서 계속해서 인식이 되고 있는 경우는 2초가 지나 문이 닫히면 안 되기 때문에 TIM4와 TIM1을 각각 계속해서 초기화 하는 코드까지 추가해주었다.



사진을 통해서 정확한 동작을 묘사하기는 힘들지만 위 사진들과 같이 시뮬레이션 상에서 PA10, PA11, PA12 버튼을 눌러가면서 PC0 – PC7의 ODR을 통해 출력되는 신호를 확인할 수 있었고 또한 watch window에서 left_door와 right_door에 저장된 값을 통해서도 문의 상태를 파악할 수 있었다.

이를 통해서 PA10, PA11, PA12 신호를 통해 입력을 받아 해당 입력에 따라 좌, 우측 자동문이 열리고 타이머를 이용해서 문이 닫히는 프로그램을 가상으로 만들어 보았다. 그 과정에서 하나의 인터럽트 핸들러에서 어떤 원인에 의해 인터럽트가 발생하였는지 파악해서 각종 flag 변수를 통해 상태를 독립적으로 관리하고 동작시켜 보면서 GPIO, TIMER, INTERRUPT에 대한 이해를 높일 수 있었다.