

강의명 : 마이크로프로세서

실습 번호 : 6

실습 제목 : SysTick timer exception programming

학생 이름 : 이학민

학번 : 201910906

1. SysTick timer 예외 프로그램

1.1

void SysTick_init(void)

```
{  
    *(volatile unsigned int *) SYST_RVR = ((120*1000000/1000)-1);  
    *(volatile unsigned int *) SYST_CVR = 0;  
    *(volatile unsigned int *) SYST_CSR = 0x4;  
}
```

<프로그램 설명>

SysTick_init() 함수는 레지스터 SYST_RVR, SYST_CVR, SYST_CSR을 사용하여 processor clock을 사용하는 1000Hz timer를 설정하고 timer를 disable하는 함수이다. 즉 initialize하는 함수이다. initialize할 때, 1000Hz timer 설정을 해야 한다. SYST_RVR 레지스터는 time reload 값을 설정하는 기능을 수행하는데 1/1000초 마다 정확하게 interrupt이 발생하게 만들려면 이 레지스터의 값을 (processor clock)/1000 - 1로 설정해야 한다. 실습에서 사용된 processor clock의 값은 120MHz이므로 SYST_RVR 레지스터에 (120*1000000/1000)-1을 대입한다. SYST_CVR 레지스터는 timer의 현재 값을 담는 레지스터이다. initialize해야 하므로 이 레지스터의 값을 0으로 초기화한다. SYST_CSR 레지스터는 timer 제어 및 상태 표시의 기능을 가진다. SYST_CSR 레지스터의 Bit[2]는 clock의 source를 표시한다. 이 값이 0일 때는 IMPLEMENTATION DEFINED external reference clock이고 1일 때는 processor clock이다. Bit[1]은 timer가 0에 도달하였을 때 SysTick exception의 발생을 설정하는 비트로 no effect일 때는 0의 값을 가지고 pending일 때는 1의 값을 가진다. Bit[0]은 timer를 enable 또는 disable 시키는 비트로 0일 때는 disable, 1일 때는 enable이다. 본 실습에서는 processor clock를 사용하였으므로 Bit[2]=1이 된다. timer disable시에는 Bit[1]=0, Bit[0]=0 으로 설정되어야 하므로 16진수 4, 즉 0x4를 SYST_CSR 레지스터의 값에 대입한다. *(volatile unsigned int *)를 레지스터 변수 앞에 붙인 이유는 형변환을 하기 위함이다. unsigned integer constant형 변수를 unsigned integer pointer형으로 강제 형변환하기 위해 (volatile unsigned int *)를 사용하였고 그 주소에 해당되는 값에 접근하기 위해 맨 앞에 *를 한 번 더 붙여주었다.

```
void SysTick_enable(void)
{
    *(volatile unsigned int *) SYST_CSR = 0x7;
}
```

<프로그램 설명>

SysTick_enable() 함수는 레지스터 SYST_CSR을 사용하여 timer를 enable 하는 역할을 수행한다. SYST_CSR 레지스터는 timer 제어 및 상태 표시의 기능을 가진다. SYST_CSR 레지스터의 Bit[0], Bit[1], Bit[2]에 대한 상세한 설명은 앞서 설명하였다. timer enable시에는 Bit[2]=1, Bit[1]=1, Bit[0]=1 으로 설정되어야 하므로 16진수 7, 즉 0x7를 SYST_CSR 레지스터의 값에 대입한다. *(volatile unsigned int *)를 레지스터 변수 앞에 붙인 이유는 형변환을 하기 위함이다. unsigned integer constant형 변수를 unsigned integer pointer형으로 강제 형변환하기 위해 (volatile unsigned int *)를 사용하였고 그 주소에 해당되는 값에 접근하기 위해 맨 앞에 *를 한 번 더 붙여주었다.

```
void SysTick_disable(void)
{
    *(volatile unsigned int *) SYST_CSR = 0x4;
}
```

<프로그램 설명>

SysTick_disable() 함수는 레지스터 SYST_CSR을 사용하여 timer를 disable 하는 역할을 수행한다. SYST_CSR 레지스터는 timer 제어 및 상태 표시의 기능을 가진다. SYST_CSR 레지스터의 Bit[0], Bit[1], Bit[2]에 대한 상세한 설명은 앞서 설명하였다. timer disable시에는 Bit[2]=1, Bit[1]=0, Bit[0]=0 으로 설정되어야 하므로 16진수 4, 즉 0x4를 SYST_CSR 레지스터의 값에 대입한다. *(volatile unsigned int *)를 레지스터 변수 앞에 붙인 이유는 형변환을 하기 위함이다. unsigned integer constant형 변수를 unsigned integer pointer형으로 강제 형변환하기 위해 (volatile unsigned int *)를 사용하였고 그 주소에 해당되는 값에 접근하기 위해 맨 앞에 *를 한 번 더 붙여주었다.

```
void SysTick_Handler(void)
{
    tick += 1;
}
```

<프로그램 설명>

SysTick_Handler() 함수는 SysTick timer exception의 handler 함수로, 현재 시간을 표시하는 전역 변수 tick을 1 증가시키는 함수이다. 이때, 1000Hz timer이므로 1tick은 1/1000 sec의 값을 갖는다.

2. SysTick timer 예외 프로그램 시험

2.1

```
#include "frdm_k64f.h"          // include for FRDM-K64F board
#include "systick.h"

unsigned int tick = 0;

int main(void)
{
    char input[80];

    soc_init();                  // initialize FRDM-K64F board

    for (;;) {
        printf("=====\n");
        printf("SysTick exception\n");
        printf("=====\n");
        printf("1. Init 1,000Hz SysTick timer\n");
        printf("2. Enable SysTick timer\n");
        printf("3. Disable SysTick timer\n");
        printf("4. Print the current tick\n\n");
        printf("Select one: ");

        gets(input);
        printf("\n");
        if ((input[0] == '1') && (input[1] == '\0')) {
            tick = 0;
            printf("tick=%d\n", tick);
            SysTick_init();
        } else if ((input[0] == '2') && (input[1] == '\0')) {
            SysTick_enable();
            printf("tick=%d\n", tick);
        } else if ((input[0] == '3') && (input[1] == '\0')) {
            SysTick_disable();
            printf("tick=%d\n", tick);
        } else if ((input[0] == '4') && (input[1] == '\0')) {
            printf("tick=%d\n", tick);
        } else {
            printf("tick=%d\n", tick);
        }
    }
}
```

```

    }
    printf("\n");
}

return 0;
}

```

<main() 함수 분석>

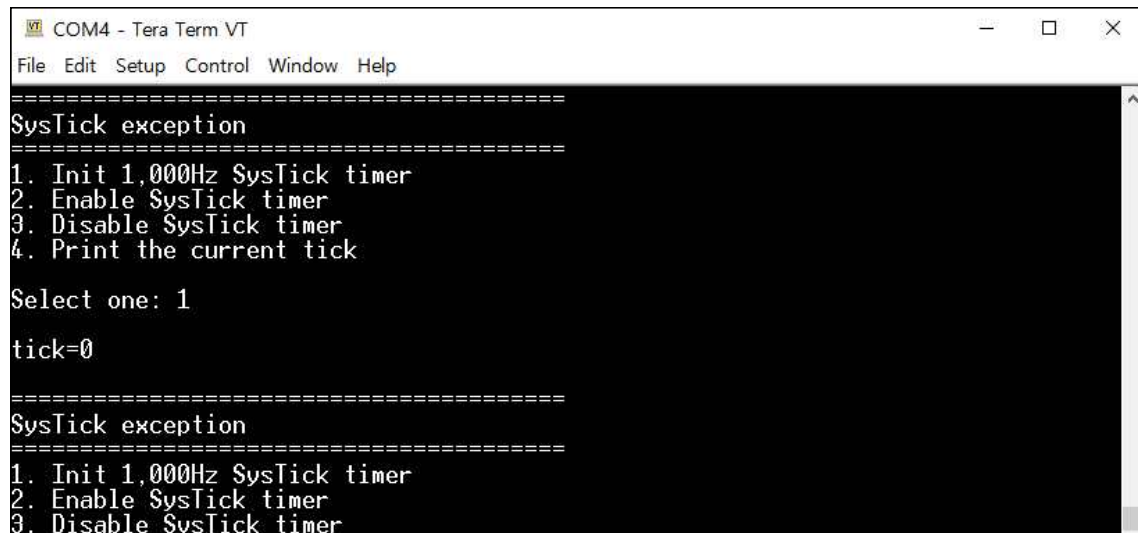
문자열을 받을 수 있는 크기가 80인 char형 배열 input[]을 선언한다. soc_init()를 통해 보드를 초기화한 후 다음 동작을 수행한다. 사용자에게 선택지를 안내해주는 문구를 출력한다. 간단한 출력 예시는 다음과 같다.

1. 1,000Hz SysTick timer
 2. Enable SysTick timer
 3. Disable SysTick timer
 4. Print the current tick
- Select one:

사용자가 안내 문구를 보고 1~4 사이의 숫자를 입력하여 그에 알맞은 동작을 수행하도록 조건문을 설계한다. 사용자가 원하는 숫자는 gets 함수를 통해 입력받는다. 1부터 4 사이의 임의의 숫자 n을 입력했을 때, n의 값은 input[0]와 대응되고 문자열을 모두 입력받고 나면 NULL문자가 문자열 맨 끝에 생기므로 input[1]에는 '\0'이 대응된다. 사용자가 1을 입력하였을 때는 (input[0] == '1') && (input[1] == '\0') 조건이 만족되고 이 조건이 만족되면 SysTick timer를 initialize해야 하므로 현재 시간에 해당하는 변수 tick을 0으로 초기화하고, 그 값을 화면에 출력한 후에 위에서 설명한 SysTick_init()를 불러와 initialize 과정을 수행한다. 사용자가 2를 입력하면 (input[0] == '2') && (input[1] == '\0') 조건이 만족되고 앞서 설명한 SysTick_enable()를 불러와 SysTick timer를 enable(작동)하게 한다. 또한 현재 시간의 값을 가지고 있는 변수 tick을 화면에 출력한다. 사용자가 3을 입력하면 (input[0] == '3') && (input[1] == '\0') 조건이 만족되고 앞서 설명한 SysTick_disable()을 불러와 SysTick timer를 disable하게 만든다. 또한 현재 시간의 값을 가지고 있는 변수 tick을 화면에 출력한다. 사용자가 4를 입력하거나 ((input[0] == '4') && (input[1] == '\0')) 그 밖의 경우(else)에 해당되면 현재 시간을 화면에 출력하는 코드를 수행한다. for문으로 만든 무한루프이기 때문에 이 과정을 계속 반복한다. 무한루프를 벗어날 시에는 0을 반환하며 main함수를 종료한다.

2.2

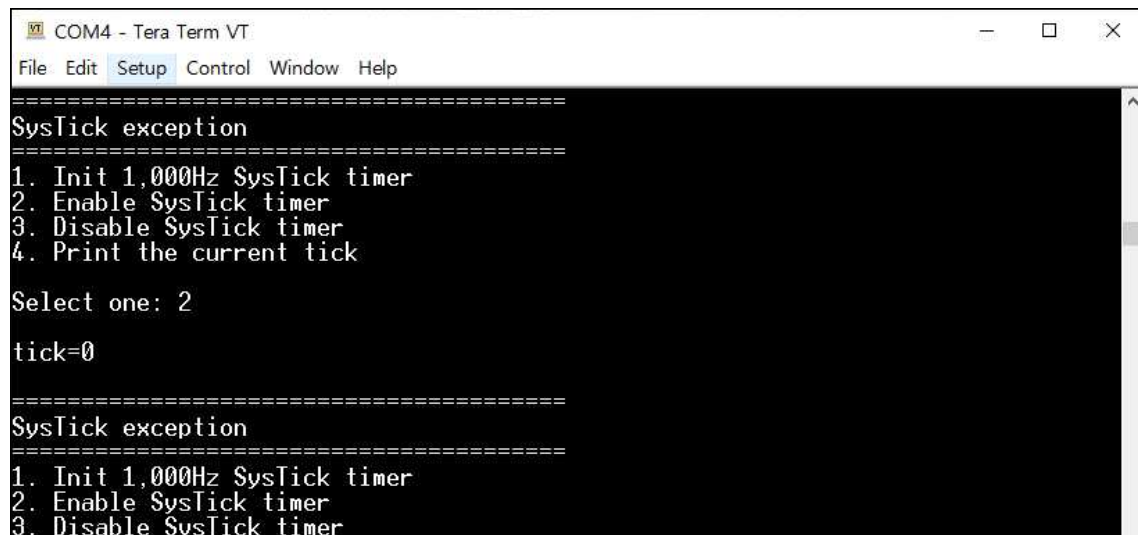
<동작 1>



```
COM4 - Tera Term VT
File Edit Setup Control Window Help
=====
SysTick exception
=====
1. Init 1,000Hz SysTick timer
2. Enable SysTick timer
3. Disable SysTick timer
4. Print the current tick
Select one: 1
tick=0
=====
SysTick exception
=====
1. Init 1,000Hz SysTick timer
2. Enable SysTick timer
3. Disable SysTick timer
```

main 함수에서 사용자가 1을 선택하였을 때 tick을 0으로 초기화하고 그 tick 값을 printf를 통해 출력하도록 했으므로 tick=0이 출력된다. tick=0이 출력된 후에 앞서 설명한 SysTick_init()을 호출하여 timer를 initialize한다.

<동작 2>



```
COM4 - Tera Term VT
File Edit Setup Control Window Help
=====
SysTick exception
=====
1. Init 1,000Hz SysTick timer
2. Enable SysTick timer
3. Disable SysTick timer
4. Print the current tick
Select one: 2
tick=0
=====
SysTick exception
=====
1. Init 1,000Hz SysTick timer
2. Enable SysTick timer
3. Disable SysTick timer
```

사용자가 2를 선택하였을 때, main함수는 SysTick_enable()를 호출하고 timer가 작동하게 된다. SysTick_enable()이 호출되고 tick의 값이 시간에 따라 점차 증가한다. 이때 동작 1에서 알 수 있듯이 tick은 0으로 초기화했었으므로 timer가 start할 때의 tick의 값은 0이다. 따라서 tick=0이 화면에 출력된다.

<동작 3>

```
COM4 - Tera Term VT
File Edit Setup Control Window Help
=====
SysTick exception
=====
1. Init 1,000Hz SysTick timer
2. Enable SysTick timer
3. Disable SysTick timer
4. Print the current tick

Select one: 4

tick=8764

=====
SysTick exception
=====
1. Init 1,000Hz SysTick timer
2. Enable SysTick timer
3. Disable SysTick timer
```

동작 2를 통해 timer를 작동시켰다. timer가 start되었으므로 tick의 값이 시간이 지남에 따라 점차적으로 늘어난다. 본 실습에서 이용한 timer는 1000Hz이므로 0.001초가 흐를 때마다 tick의 값이 1씩 증가한다. 동작 2를 수행하며 timer가 시작되어 tick의 값이 늘어난다. 동작 3을 수행하면 현재 tick의 값을 화면에 출력하여 사용자가 볼 수 있는데 tick의 값은 timer가 시작된 순간부터 얼마만큼의 시간이 흘렀는지를 나타낸다. 따라서 동작 2와 동작 3 사이에 걸린 시간을 나타낸다고 할 수 있고 이 시간이 총 8.764초 였으므로 총 증가한 tick의 값은 8764가 된다.

<동작 4>

```
COM4 - Tera Term VT
File Edit Setup Control Window Help
=====
SysTick exception
=====
1. Init 1,000Hz SysTick timer
2. Enable SysTick timer
3. Disable SysTick timer
4. Print the current tick

Select one: 3

tick=25930

=====
SysTick exception
=====
1. Init 1,000Hz SysTick timer
2. Enable SysTick timer
3. Disable SysTick timer
```

동작 4를 수행하면 SysTick_disable()을 호출하여 timer를 중지시킨다. timer가 중지되었으므로 더 이상 tick의 값은 늘어나지 않는다. 동작 4에서 timer가 중지되기 전까지 동작 2에서부터 증가하기 시작한 tick의 값은 꾸준히 증가했을 것이다. 동작 4의 화면에서 출력되는 tick의 값은 timer가 시작된 때부터 timer가 중지된 때까지 걸린 시간을 나타낸다. 이는 동작 2와 동작 4 사이에 걸린 시간이고 총 25.930초가 걸렸으므로 tick의 값은 25930이 된다.

<동작 5>

```
COM4 - Tera Term VT
File Edit Setup Control Window Help
=====
SysTick exception
=====
1. Init 1,000Hz SysTick timer
2. Enable SysTick timer
3. Disable SysTick timer
4. Print the current tick

Select one: 4

tick=25930

=====
SysTick exception
=====
1. Init 1,000Hz SysTick timer
2. Enable SysTick timer
3. Disable SysTick timer
```

앞서 동작 4에서 SysTick_disable()를 호출하여 timer를 중지시켰으므로 더 이상 tick의 값은 늘어나지 않는다. 따라서 동작 4에서 출력된 tick의 값이 계속 유지된다. 따라서 동작 5에서의 tick 값 또한 25930이다.

<동작 6>

```
COM4 - Tera Term VT
File Edit Setup Control Window Help
=====
SysTick exception
=====
1. Init 1,000Hz SysTick timer
2. Enable SysTick timer
3. Disable SysTick timer
4. Print the current tick

Select one: 2

tick=25930

=====
SysTick exception
=====
1. Init 1,000Hz SysTick timer
2. Enable SysTick timer
3. Disable SysTick timer
```

동작 6에서는 SysTick_enable()을 호출하여 중지 상태였던 timer를 다시 작동하게 한다. 다시 tick의 값이 시간에 따라 점차 증가한다. 이때 동작 5에서 마지막으로 저장되어있던 tick의 값은 25930이므로 timer가 다시 작동할 때의 tick의 값은 25930이다. 따라서 tick=25930이 화면에 출력된다.

<동작 7>

```
COM4 - Tera Term VT
File Edit Setup Control Window Help
=====
SysTick exception
=====
1. Init 1,000Hz SysTick timer
2. Enable SysTick timer
3. Disable SysTick timer
4. Print the current tick

Select one: 4

tick=34658

=====
SysTick exception
=====
1. Init 1,000Hz SysTick timer
2. Enable SysTick timer
3. Disable SysTick timer
```

동작 7에서는 시간에 따라 증가하는 tick의 현재값을 화면에 출력한다. timer가 최초로 시작된 동작 2에서부터 동작 7까지 걸린 총 시간이 변수 tick에 담겨 있어야 하지만 동작 4에서 timer를 잠시 중지시켰으므로 동작 4부터 동작 6이 수행되기 직전까지는 tick의 값이 증가하지 않았다. 따라서 동작 7에서 출력되는 tick의 값은 동작 2에서 동작 4 사이에 걸린 시간과 동작 6에서 동작 7 사이에 걸린 시간의 합을 나타낸다. 이는 동작 5에서 출력한 tick의 값에 동작 6과 동작 7 사이에 걸린 tick을 더한 값과 같다. 동작 6과 동작 7 사이에 8.728초가 소요되었고 바꾸어 말하면 tick은 8728 증가하였으므로 동작 7에서는 tick=34658이 출력된다.

끝.