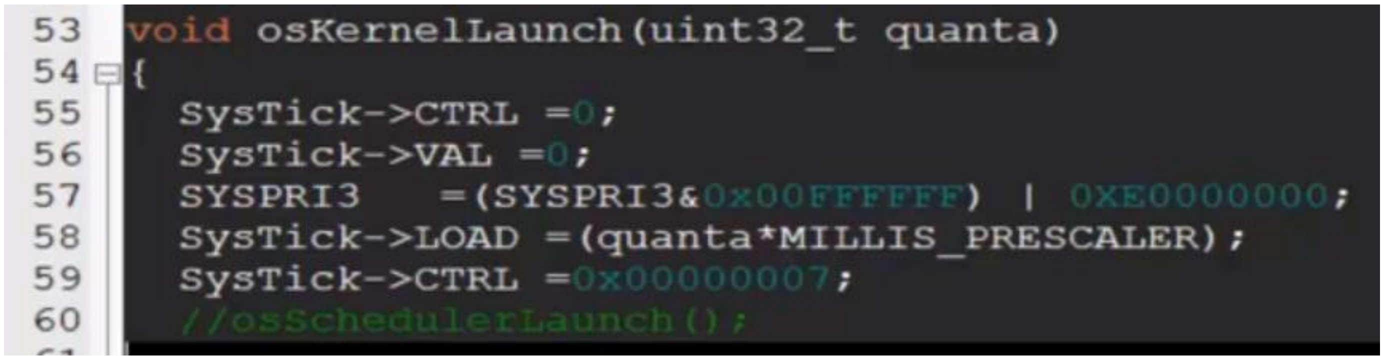
Problem of setting the systick priority **SYSPRI3 = (SYSPRI3 & 0x00FFFFFF) | 0xE0000000; // Priority 7**

SYSPRI3 =(SYSPRI3&0x00FFFFFF)|0xE0000000; // priority 7      #define SYSPRI3 (*((volatile uint32_t *)0xE000ED20))



```
void osKernelLaunch(uint32_t quanta)
{
  SysTick->CTRL =0;
  SysTick->VAL =0;
  SYSPRI3    =(SYSPRI3&0x00FFFFFF) | 0XE0000000;
  SysTick->LOAD =(quanta*MILLIS_PRESCALER);
  SysTick->CTRL =0x00000007;
  //osSchedulerLaunch();
```

The instructor reply on this problem in the below image.

You can find the register we renamed to `SYSPRI3` in the Cortex-M4 Generic User Guide page **238**. This register is called

`System Handler Priority Register 3` in that document.

When you take a look at the register diagram you realize that **bits24 -31** is named `PRI_15` and when you read its function below the diagram it says *"Priority of system handler 15, Systick Exception"*- This is the register for setting the priority of Systick.

As you can see the ARM Cortex-M4 design allows 0-255 priority levels. We know this because **bit24 to bit31** = **8bits**

**2^8 = 256**

We count from zero, therefore we have **0-255**

Now, this is what cortex-m4 design allow however silicon manufacturers such as STMicroelectronics and Texas Instruments etc. do not need to use all the 0-255 levels.

Our microcontroller, STM32 and TM4C123 uses **0-7** levels of priority. Zero being the highest priority and 7 being the lowest priority.

To set this up in our `SYSPRI3` register which is also known as the `System Handler Priority Register 3` in the Cortex-M4 Generic user guide we need to use only bits **29-31** of the register. Bits 29-31 = 3 bits largest 3 bits number is 7.

If we have a 32 bits register and we wish to set **1 at bit29**, **1 at bit30** and **1 at bit31** we will have to write **0xE0000000.** If you expand **hexadecimal E** to binary you get **1110**. So by writing this we set the **priority to 7.**

***Now the other part :***

`(SYSPRI3 & 0x00FFFFFF)` : This part keeps every thing in the `SYSPRI3` the same while clearing only

bits24 -31 is i.e. `PRI_15` as it is called in the Cortex-M4 Generic User Guide.

We first clear these bits and then we set them to the new priority values.

As an example if we wish to set **Systick to Priority 2**, we shall write:

```
SYSPRI3 = (SYSPRI3 & 0x00FFFFFF) | 0x40000000;
```

If we wish to set **Systick to Priority 0**, we shall write:

```
SYSPRI3 =  SYSPRI3 & 0x00FFFFFF ;
```

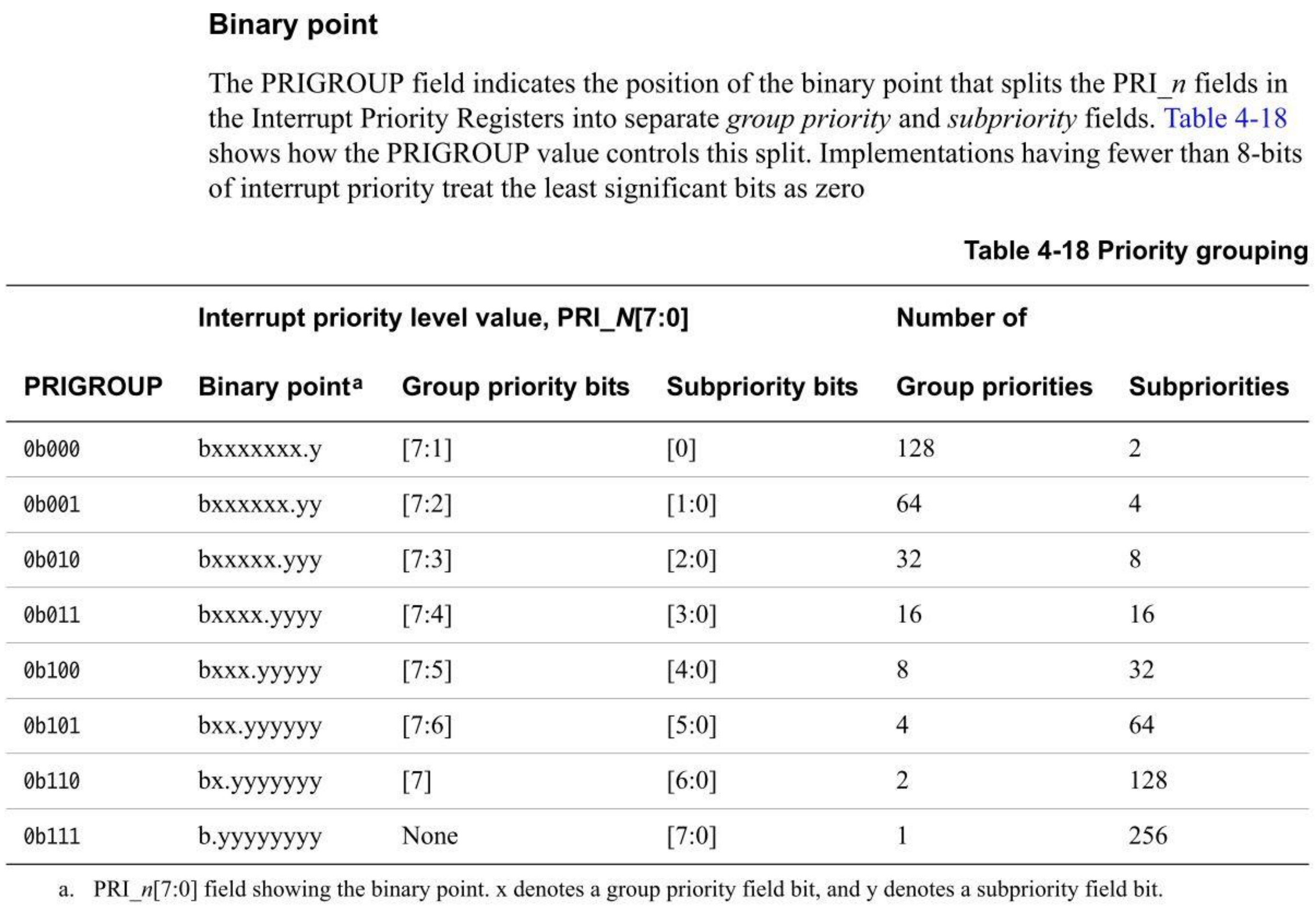
According to my research, I think the explanation is not 100% correct. You mix the Priority Grouping with the priority.

1. Cortex Arm M3 M4 processors use 8 bits to store the priorities for preemption and sub priorities. (http://micromouseusa.com/?p=279#comment-567026)

2. There are 8 ways (0 ~ 7) to set up the 8 bits for the priorities by using

void NVIC_SetPriorityGrouping(uint32_t PriorityGroup).

The PriorityGroup value can be found below. It tells you how many bits are assigned for Group priority and how many for subpriority. The default is 0 (https://www.keil.com/pack/doc/CMSIS/Core/html/group__NVIC__gr.html#gad78f447e891789b4d8f2e5b21eeda354). This is not for setting the priority.



**Binary point**

The PRIGROUP field indicates the position of the binary point that splits the PRI_*n* fields in the Interrupt Priority Registers into separate *group priority* and *subpriority* fields. Table 4-18 shows how the PRIGROUP value controls this split. Implementations having fewer than 8-bits of interrupt priority treat the least significant bits as zero

**Table 4-18 Priority grouping**

| | Interrupt priority level value, PRI_*N*[7:0] | | | Number of | |
|---|---|---|---|---|---|
| **PRIGROUP** | **Binary point**[a] | **Group priority bits** | **Subpriority bits** | **Group priorities** | **Subpriorities** |
| 0b000 | bxxxxxxx.y | [7:1] | [0] | 128 | 2 |
| 0b001 | bxxxxxx.yy | [7:2] | [1:0] | 64 | 4 |
| 0b010 | bxxxxx.yyy | [7:3] | [2:0] | 32 | 8 |
| 0b011 | bxxxx.yyyy | [7:4] | [3:0] | 16 | 16 |
| 0b100 | bxxx.yyyyy | [7:5] | [4:0] | 8 | 32 |
| 0b101 | bxx.yyyyyy | [7:6] | [5:0] | 4 | 64 |
| 0b110 | bx.yyyyyyy | [7] | [6:0] | 2 | 128 |
| 0b111 | b.yyyyyyyy | None | [7:0] | 1 | 256 |

a. PRI_*n*[7:0] field showing the binary point. x denotes a group priority field bit, and y denotes a subpriority field bit.

**Register **AIRCR** who govern the grouping behavior

**Table 4-12 Summary of the system control block registers**

| Address | Name | Type | Required privilege | Reset value | Description |
|---|---|---|---|---|---|
| `0xE000E008` | ACTLR | RW | Privileged | `0x00000000` | *Auxiliary Control Register* |
| `0xE000ED00` | CPUID | RO | Privileged | `0x410FC240` | *CPUID Base Register* on page 4-13 |
| `0xE000ED04` | ICSR | RW[a] | Privileged | `0x00000000` | *Interrupt Control and State Register* on page 4-13 |
| `0xE000ED08` | VTOR | RW | Privileged | `0x00000000` | *Vector Table Offset Register* on page 4-16 |
| `0xE000ED0C` | AIRCR | RW[a] | Privileged | `0xFA050000` | *Application Interrupt and Reset Control Register* on page 4-16 |
| `0xE000ED10` | SCR | RW | Privileged | `0x00000000` | *System Control Register* on page 4-19 |
| `0xE000ED14` | CCR | RW | Privileged | `0x00000200` | *Configuration and Control Register* on page 4-19 |
| `0xE000ED18` | SHPR1 | RW | Privileged | `0x00000000` | *System Handler Priority Register 1* on page 4-21 |
| `0xE000ED1C` | SHPR2 | RW | Privileged | `0x00000000` | *System Handler Priority Register 2* on page 4-22 |
| `0xE000ED20` | SHPR3 | RW | Privileged | `0x00000000` | *System Handler Priority Register 3* on page 4-22 |