3 Section: Developing a StreamBuffer USB virtual COM port (pg 300)

- **Clarification**, page 300
  - `VirtualComDriver.c` includes the function `GetUsbRxStreamBuff()` and the stream-buffer `vcom_rxStream`. They aren't used until Chapter 13.

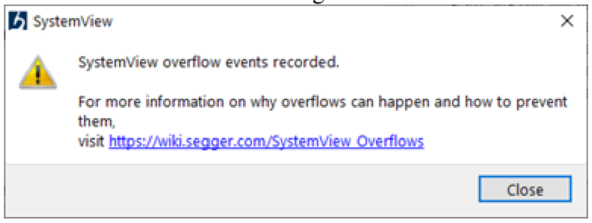
- **Bug in book and code** (`VirtualComDriver.c`), page 300
  - `TransmitUsbDataLossy()` uses `xStreamBufferSendFromISR()`, but it should use `xStreamBufferSend()`, with the parameter `xTicksToWait` set to 0.

```
int32_t numBytesCopied = xStreamBufferSend( txStream, Buff, Len, 0);
```

  - `TransmitUsbDataLossy()` is called from a task, not from an ISR.
  - From the FreeRTOS manual: "*Use xStreamBufferSend() to write to a stream buffer from a task. Use xStreamBufferSendFromISR() to write to a stream buffer from an interrupt service routine (ISR).*"

- **Clarification**, page 300
  - `VirtualCommInit()` calls `MX_USB_DEVICE_Init()`.
  - `MX_USB_DEVICE_Init()` was described earlier, on page 294. It is used to initialize the entire USB device driver stack. I think `MX_USB_DEVICE_Init()` was generated by STM32CubeMX.

- **Bug in book and code** (`VirtualCommDriver.c`, `mainUsbStreamBuffer.c`), page 308
  - **Problem**:
    - The CPU usage cannot be measured accurately due to overflows in SystemView.


SystemView

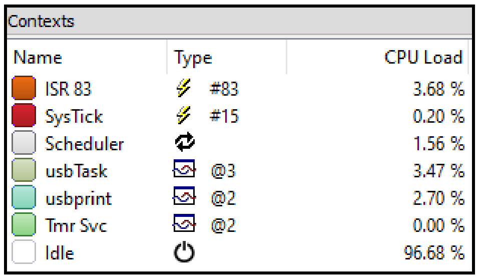
SystemView overflow events recorded.

For more information on why overflows can happen and how to prevent them,
visit https://wiki.segger.com/SystemView_Overflows

  - **Solution**:
    - SystemView ran without overflow, after applying these fixes
      - Remove calls to `SEGGER_SYSVIEW_PrintfHost()`
      - Turn-off the Ozone debugger before starting SystemView
        - Use Ozone to load the program and start it
        - Turn off the debugger via: `Debug` : `Stop Debug Session`

- **Additional info**, page 308
  - From my tests, the CPU usage is a bit different than what is shown in the book. Though, my tests used the fixes described above, to avoid overflow in SystemView.
  - My test, using the book's original code:

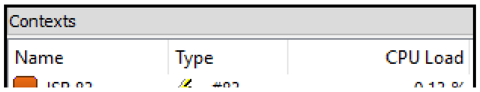
```
txStream = xStreamBufferCreate( txBuffLen, 1);
uint8_t numBytes = xStreamBufferReceive(txStream, usbTxBuff, txBuffLen,
portMAX_DELAY);
```


Contexts

Name	Type	CPU Load
ISR 83	#83	3.68 %
SysTick	#15	0.20 %
Scheduler		1.56 %
usbTask	@3	3.47 %
usbprint	@2	2.70 %
Tmr Svc	@2	0.00 %
Idle		96.68 %

  - My test, using these modifications described in the book:

```
txStream = xStreamBufferCreate(txBuffLen, 500);
uint8_t numBytes = xStreamBufferReceive(txStream, usbTxBuff, txBuffLen, 100);
```


Contexts

Name	Type	CPU Load
[illegible]	[illegible]	[illegible]