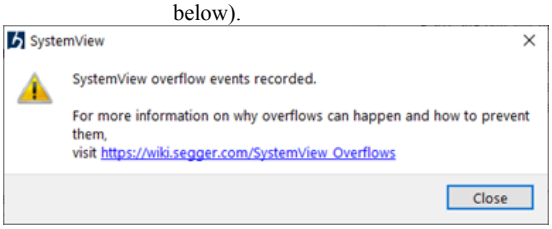
- **Bug in the code** (`mainUartPolled.c`), page 241
  - **Problem**:
    - When I ran the code, SystemView did not display the messages generated by the calls to `SEGGER_SYSVIEW_PrintfHost()`.
  - **Analysis**:
    - Context:
      - For the `.jdebug` file, I had applied the fixes described above.
      - The SystemView Recorder was started after the scheduler started.
    - One hypothesis is that the polling process was generating so many events that it was overwhelming the debugger and SystemView, when the SystemView Recorder was started.
  - **Solution**:
    - I added code that allowed the SystemView Recorder to be started after the scheduler started, but before the `polledUartReceive()` task was run.
    - This code was added to the beginning of `uartPrintOutTask()`:

```
RedLed.On();
while(!ReadPushButton());
```

    - How to run the program:
      - Run the program on the board via Ozone.
      - Start the SystemView Recorder after the red LED is on.
      - After the Recorder is started, press the board's blue push-button
    - The messages from `SEGGER_SYSVIEW_PrintfHost()` now get displayed. However, SystemView also displays a message indicating overflow has occurred. That problem is described next.

- **Bug in the code** (`mainUartPolled.c`), page 241
  - **Problem**:
    - To demonstrate the UARTs' data-transfer is working, each received byte is displayed using `SEGGER_SYSVIEW_PrintfHost()`. However, for many of the calls to `SEGGER_SYSVIEW_PrintfHost()`, the data is not displayed due to problems in SystemView.
    - When SystemView Recorder is run, SystemView displays a message stating that overflow has occurred, and that SystemView is unable to record all generated events (screen-shot below).



    - The problem is that, with the overflow, it's not possible to know if missing messages are due to the overflow, or due to UART-related errors. The messages don't demonstrate whether the UARTs' data-transfer is working.
  - **Background**:
    - `UART-4` sends the string defined in `uart4Msg[]` in `Uart4Setup.c`. The string's bytes are sent serially and continuously.
    - The bytes are received by the task `polledUartReceive()`, in `mainUartPolled.c`. The task `uartPrintOutTask()` writes each byte, one at a time, using `SEGGER_SYSVIEW_PrintfHost()`.
  - **Solution**:
    - One way to solve the problem is to collect stats on the data received. The stats are displayed infrequently to reduce the overflow, e.g., every second. The stats are:
      - Counts of properly-received strings, i.e., the chars in `uart4Msg[]` are all received and in order.
      - Counts of chars received that are not among the chars in `uart4Msg[]`.
    - An implementation of the solution is here: https://github.com/jimyuill/embedded-systems-projects-01/tree/main/book--Hands-On-RTOS/chapter-10--mainUartPolled--fixed
    - Running the implemented solution showed the bytes are being transferred with little or no data loss
  - **Additional info:**
    - For fixing SystemView overflow, general techniques are presented in the study-guide's SystemView page.

- **Clarification**, page 241
  - In `uartPrintOutTask()`, the bytes received are recorded using:
    - `SEGGER_SYSVIEW_PrintfHost("%c", nextByte);`
  - For the text-string that is sent by the UART, it's null-terminator is also sent. When the null-terminator is displayed by `SEGGER_SYSVIEW_PrintfHost()`, just an empty string is shown on the SystemView app.