Problem:

While establishing UART communication between two PSoC64 boards, whenever a char is received, PSoC64 throws a fault. This happens only when we use interrupts on the receiver side. If polling approach is used the serial communication works fine.

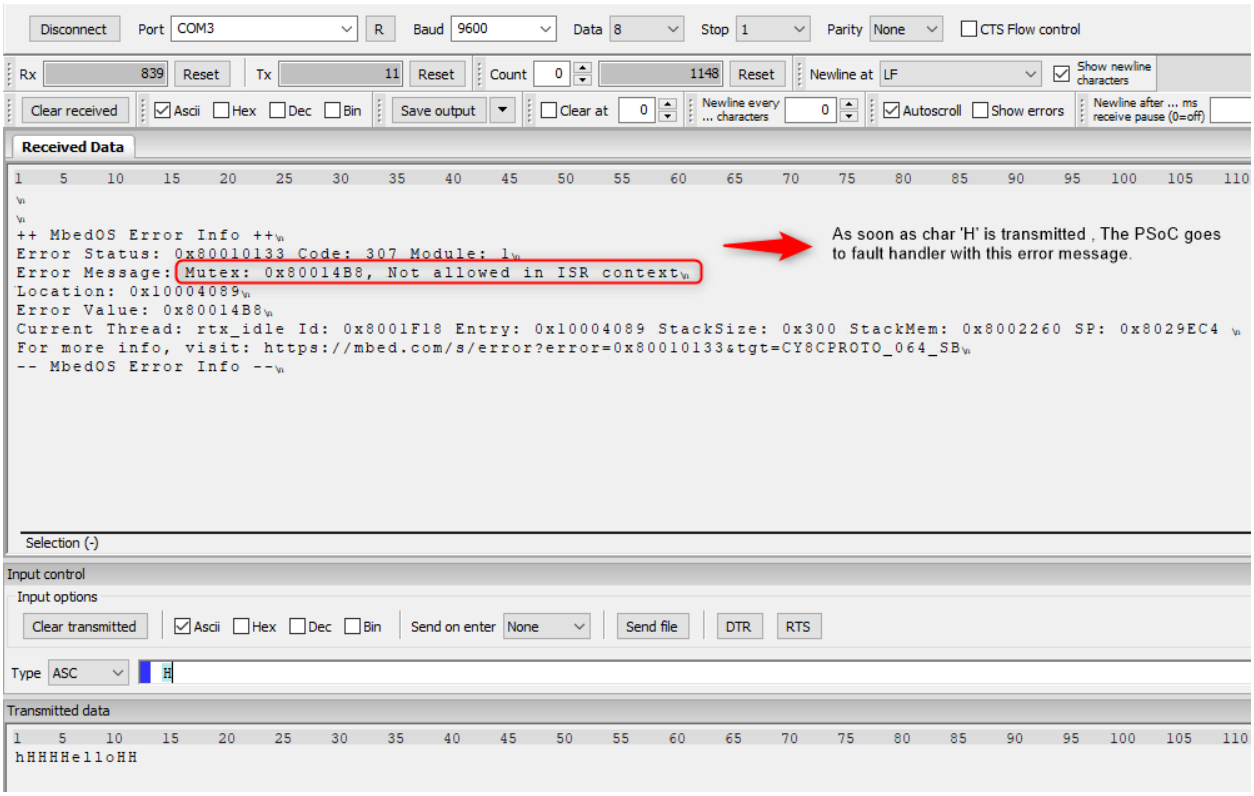
Description:

As part of project, we were trying to establish a secure communication between two devices running PSoC64 MCU. For communication, we used UART protocol.

In order to receive data without any loss, we used interrupts at the receiver PSoC64 board. We have used Mbed OS in our project version 5.14. So for establishing UART communication with PSoC64 , we use the abstraction class (we tried Serial and UARTSerial class) provided by Mbed-OS , which internally uses HAL driver of PSoC64 MCU.

But whenever we receive data , the PSoC64 goes to the fault handler. See the screenshot attached.

Screenshot:



Code:

The application code that we used is attached in the zip file as main_usingInterrupt.cpp

The explanation we found:

Mbed OS is a secure OS with thread safety support. Execution of certain function in an ISR context is not safe. For example, printf() in an interrupt context causes a Mutex error.

Our Assumption on why this would have occurred ?

We assume that the reason being, statements like printf() are blocking call , hence it uses CPU time , it has to wait in the ISR till it is executed. Meanwhile if any interrupt comes it will not be serviced. Therefore, to enhance the real time behavior they might have prohibited usage of any blocking or such functions.

Still to ponder:

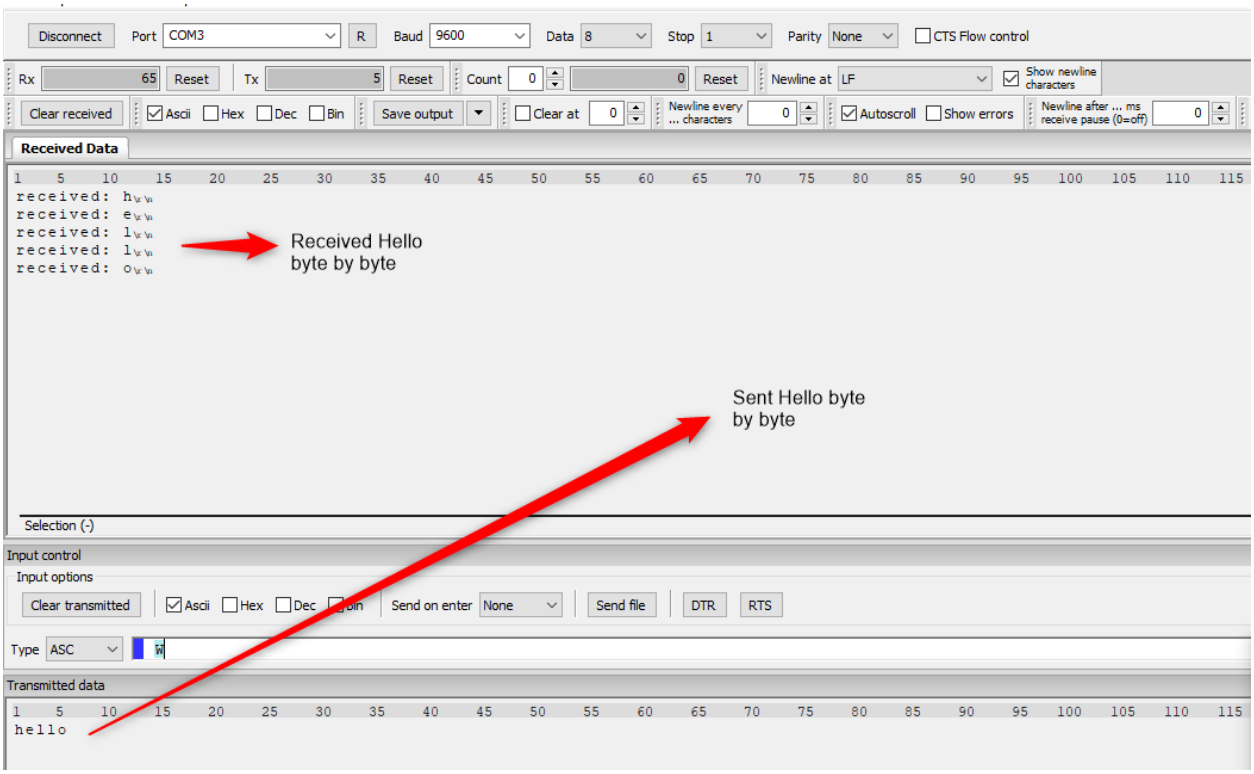
However, we have not used any printf in our ISR, we are simply reading byte by byte using getc function. Still we are getting the fault.

Possible way out :

To be able to execute code in the intended behavior they have provided EventQueue Mechanism. Event Queue Using defers execution of code from an interrupt context to a user context. More about EventQueue: https://os.mbed.com/docs/mbed-os/v5.15/tutorials/the-eventqueue-api.html

Solution using EventQueue:

We were able to run the code using eventqueue, please find the screenshot below :



Please find the code for Event Queues here: main_usingEventQueue.cpp