UART2 Transmit Setup

Configure UART2 for 9600 baud rate, 8 data bits, no parity, normal (not inverted) assertion levels. Use pin PTD3 for transmit data out from the MCU.

CMSIS names shown for registers and accessors — page number in reference [2] unless noted as ref [1]

Create an initialization function to do the following:

1) Turn on UART2 and PORTD bus clocks
   - Registers: set bits in SIM->SCGC4 and SIM->SCGC5
   - Accessors: SIM_SCGC4_UART2_MASK — 205
     SIM_SCGC5_PORTD_MASK — 206

2) Configure PORTD pin 3 for UART2 output
   - Registers: PORTD->PCR[3],
   - Accessors: ~PORT_PCR_MUX_MASK to clear the mux bits
     PORT_PCR_MUX(x) where x is the desired alt function — 47[1]

3) Make sure UART2 transmit, receive, and interrupt causes are turned off
   - Register: UART2->C2 — 753
     clear all bits by setting this register to zero

4) Establish desired buad rate. A 13 bit prescale divider creates a clock that is 16 times faster than the desired baud rate. Input to the prescaler is the UART module clock which is the bus clock. If CLOCK_SETUP 1 is being used then the bus clock is 24 Mhz. The required 13 bit divisor value is named SBR[12:0] (see diagram on page 762 [2]). 8-bit registers are used in the UART circuit, so the 13 bit prescale value is split with 5 bits in a high register and 8 in a low register.
   ```
   sbr= (uint16_t)((24000000)/(baud_rate*16));
   ```
   - Registers: UART2->BDH is the high register — 751
     UART2->BDL is the low register — 752
   - Action: assign sbr>>8 to BDH
     assign sbr to BDL (upper bits will be ignored)
   - Note: it is ok in this step to write zeros to the upper 3 bits of BDH

5) Configure control register for 8 data bits, no parity, one stop bit
   - Register: UART2->C1
     clear all bits by setting this register to zero

6) Enable transmit
   - Register: UART2->C2
   - Action: set bit3 to enable transmitter. Use MASK(3)

(see next page)

Programming considerations

Assuming that UART2 is properly initialized, transmitting a character is achieved by writing a character to the data holding register, register D in UART2, i.e. UART2->D. After writing to the data holding register, which initiates data transmission, you need to read from register UART2->S1, check bit 7, and not write another character to UART2->D until bit 7 becomes a 1. . Bit 7 will be set, i.e. become logic 1, when the holding register becomes empty because its contents was moved to the shift register. Another character can then be placed in the holding register.

To test the initialization of UART2 write a program that calls the initialization function once and then enters a loop to continuously display the character on an oscilloscope. Here is a suitable loop for testing UART2 initialization and operation:

```
while(1) {
    UART->D = 0x_ _;     // Insert the numeric value for a desired ASCII character
    while (!(UART->S1 & 0x80)) { }   // wait for bit to be set
    delayMs(2);              // delay to create a gap between characters sent.  Easier to see
                             //  on the scope where one character ends and another begins.
}
```

If the UART prescaler is set up correctly for 9600 baud, bit time will be about 100us long and transmission of the 8 data bits, one start bit, and one stop bit will take about 1 mSec.