```c
/*******************************************************************************
 Module
   UART_Service.c

 Description
   Manages UART Communication to XBEE

*******************************************************************************/
/*---------------------------- Include Files ----------------------------*/
/* include header files for this state machine as well as any machines at the
   next lower level in the hierarchy that are sub-machines to this machine
*/
#include "ES_Configure.h"
#include "ES_Framework.h"
#include "Master_SM.h"
#include "Definitions.h"
#include "BITDEFS.h"
#include "Helpers.h"
#include "Transmit_Service.h"
#include "Receive_SM.h"

/*---------------------------- Module Defines ----------------------------*/

/*---------------------------- Module Functions ----------------------------*/
/* prototypes for private functions for this service.They should be functions
   relevant to the behavior of this service
*/

static void initUART(void);

/*---------------------------- Module Variables ----------------------------*/
// with the introduction of Gen2, we need a module level Priority variable
static uint8_t MyPriority;

/*---------------------------- Module Code ----------------------------*/
/*******************************************************************************
 Function
     InitUART

 Parameters
     uint8_t : the priorty of this service

 Returns
     bool, false if error in initialization, true otherwise

 Description
     Initializes UART
*******************************************************************************/
bool InitUART_Service ( uint8_t Priority )
{
    ES_Event ThisEvent;

    MyPriority = Priority;

    //Initialize the UART
    initUART();

    // post the initial transition event
    ThisEvent.EventType = ES_INIT;
    if (ES_PostToService( MyPriority, ThisEvent) == true)
    {
        return true;
```

```
    }
    else
    {
        return false;
    }
}

/*****************************************************************************
 Function
     PostUART_Service

*****************************************************************************/
bool PostUART_Service( ES_Event ThisEvent )
{
    return ES_PostToService( MyPriority, ThisEvent);
}

/*****************************************************************************
 Function
    RunUART_Service

 Parameters
   ES_Event : the event to process

 Returns
   ES_Event, ES_NO_EVENT if no error ES_ERROR otherwise

 Description
   Runs any Necessary Events when Communicating via UART

*****************************************************************************/
ES_Event RunUART_Service( ES_Event ThisEvent )
{
    ES_Event ReturnEvent;
    ReturnEvent.EventType = ES_NO_EVENT; // assume no errors

    return ReturnEvent;
}

/*****************************************************************************
 private functions
 *****************************************************************************/
//Initialization Process for UART
static void initUART(void)
{

    // Enable the clock to the UART module 1
    HWREG(SYSCTL_RCGCUART) |= SYSCTL_RCGCUART_R1;

    //Wait for the UART to be ready (p 410)
    while ((HWREG(SYSCTL_PRUART) & SYSCTL_PRUART_R1 ) != SYSCTL_PRUART_R1 )
        ;

    // Enable clock to the GPIO port C
    HWREG(SYSCTL_RCGCGPIO) |= SYSCTL_RCGCGPIO_R2;

    // Wait for GPIO port C to be ready
    while ((HWREG(SYSCTL_PRGPIO) & SYSCTL_PRGPIO_R2 ) != SYSCTL_PRGPIO_R2 )
        ;

    // Configure GPIO pins for in/out/drive-level/drivetype
    // Set GPIO PC4 and PC5 as digital
```

```
    HWREG(GPIO_PORTC_BASE + GPIO_O_DEN) |= (BIT4HI | BIT5HI);

    // Set GPIO PC4 as input (receive) and PC5 as output (transfer)
    HWREG(GPIO_PORTC_BASE + GPIO_O_DIR) |= BIT5HI;
    HWREG(GPIO_PORTC_BASE + GPIO_O_DIR) &= ~BIT4HI;

    // Configure the transfer line (PC5) as open drain, see TIVA p.676
    // HWREG(GPIO_PORTC_BASE + GPIO_O_ODR) |= BIT5HI;

    // Program pins C4, C5 as alternate functions on the GPIO to use UART (Set
                                                                        High)
    HWREG(GPIO_PORTC_BASE + GPIO_O_AFSEL) |= (BIT4HI | BIT5HI);


Tiva p.1351)
    // Mux value = 2 (p.895) offset mask to clear nibbles 4 and 5
    HWREG(GPIO_PORTC_BASE + GPIO_O_PCTL) = (HWREG(GPIO_PORTC_BASE + GPIO_O_PCTL)
          & ~0x00ff0000) + (2 << ( 4 * BitsPerNibble)) +
                                          (2 << ( 5 * BitsPerNibble));

    // Disable UART module 1
    HWREG(UART1_BASE + UART_O_CTL) &= ~UART_CTL_UARTEN;

    // Baud Rate = 9600, Integer = 260, Fraction = 27
    // Write the integer portion of the BRD (260)
    HWREG(UART1_BASE + UART_O_IBRD) = 0x0104;

    // Write the fractional portion of the BRD (27)
    HWREG(UART1_BASE + UART_O_FBRD) = 0x1B;

    // Write desired serial parameters to the UART line control
    HWREG(UART1_BASE + UART_O_LCRH) |= UART_LCRH_WLEN_8;    // Set 8-bit frame
          length and clear everything else

    // Configure UART operation
    HWREG(UART1_BASE + UART_O_CTL) |= (UART_CTL_RXE | UART_CTL_TXE |
          UART_CTL_EOT); // Enable Receive and Transmit

    // Locally enable interupts for UART receive interrupt mask (RXIM)
    HWREG(UART1_BASE + UART_O_IM) |= (UART_IM_RXIM);

    // Set NVIC enable for UART1 (see Tiva p.104)
    HWREG(NVIC_EN0) |= BIT6HI;

    // Enable UART module 1
    HWREG(UART1_BASE + UART_O_CTL) |= UART_CTL_UARTEN;

    // Make sure interrupts are enabled globally
    __enable_irq( );

    //HWREG(UART1_BASE + UART_O_DR) = 0x55;

}


/*****************************************************************************
 Function
   UART InterruptResponse

 Description
      Rx and Tx interrupt response for UART communcation with the Xbee
*****************************************************************************/
```

```
void UARTInterruptResponse(void)
{
    //RECEIVING
    //If Raw Interrupt Status Register
    if ((HWREG(UART1_BASE+UART_O_MIS)& UART_MIS_RXMIS) == UART_MIS_RXMIS)
    {
        // Clear the source of the interrupt by writing ot the Masked Interrupt
        Clear Regsiter
        HWREG(UART1_BASE + UART_O_ICR) |= UART_ICR_RXIC;

        //Post that a Byte has been received to the
        ES_Event ThisEvent;
        ThisEvent.EventType = ES_BYTE_RECEIVED;
        ThisEvent.EventParam = HWREG(UART1_BASE + UART_O_DR);
        PostReceive_SM(ThisEvent);
    }

    //TRANSMITTING
    // If the interrupt response is from Tx FIFO open
    if ((HWREG(UART1_BASE+UART_O_MIS)& UART_MIS_TXMIS) == UART_MIS_TXMIS)
    {

        HWREG(UART1_BASE + UART_O_ICR) |= UART_ICR_TXIC; //Clear the flag

        //Send the Next Byte will return true if this was the last byte
        if (sendNextByte())
        {
            HWREG(UART1_BASE + UART_O_IM) &= ~UART_IM_TXIM; //if that was the
                  last byte, disable int
        };

    }
}
/*------------------------------ Footnotes -------------------------------*/
/*------------------------------ End of file -------------------------------*/
```