Dual UART Virtual Peripheral Implementation

Application Note 39
April 2001

1.0 Introduction

The Dual UART Virtual Peripheral uses the SX communications controller to provide asynchronous data communication through two RS-232 interfaces. The Virtual Peripheral has been developed using the SX Evaluation Board and has been tested using the SX-Key interface from Parallax Inc. and the SXIDE integrated development environment from Advanced Transdata Inc.

Unlike other MCUs that add functions in the form of additional silicon, the SX Series uses its fast execution rate to emulate peripheral functions in software modules, called Virtual Peripherals. On-chip hardware peripherals are only provided for functions that cannot be performed efficiently in software, such as timers and analog comparators.

1.1 Program Description

The Dual UART Virtual Peripheral implements two UART interfaces that can run at independent baud rates. Because both UARTs operate simultaneously, data transfer is much more efficient than implementations that only handle one channel at a time.

The Dual UART Virtual Peripheral is designed to operate in a multithreaded environment driven by the real-time clock/counter (RTCC). Whenever an RTCC interrupt occurs, an interrupt service routine (ISR) is called which contains a multitasker for allocating CPU bandwidth among any Virtual Peripherals which require interrupt service. Each task is called a *thread*, and the Dual UART Virtual Peripheral is assigned to `isrThread1`. In this example, the multitasker implements 16 slots for calling threads, and four of these slots are occupied by calls to `isrThread1`. Because the Dual UART Virtual Peripheral only receives service on every fourth interrupt, most of the CPU bandwidth is available for use by other Virtual Peripherals.

Before sending a character, software must check the transmit flag for the UART to be used. If the flag is clear, a character can be sent by setting the flag and calling the `sendbyte` routine. The Virtual Peripheral also features the capability to send strings.

Calls to `isrThread1` are used to service both UARTs. The program can be modified to include one UART in `isrThread1` and the other in `isrThread2`. In this case, the jump table for the multitasker must be modified to call `isrThread2` on every fourth interrupt, like `isrThread1`.

1.2 Interrupt Service Routine Flowchart

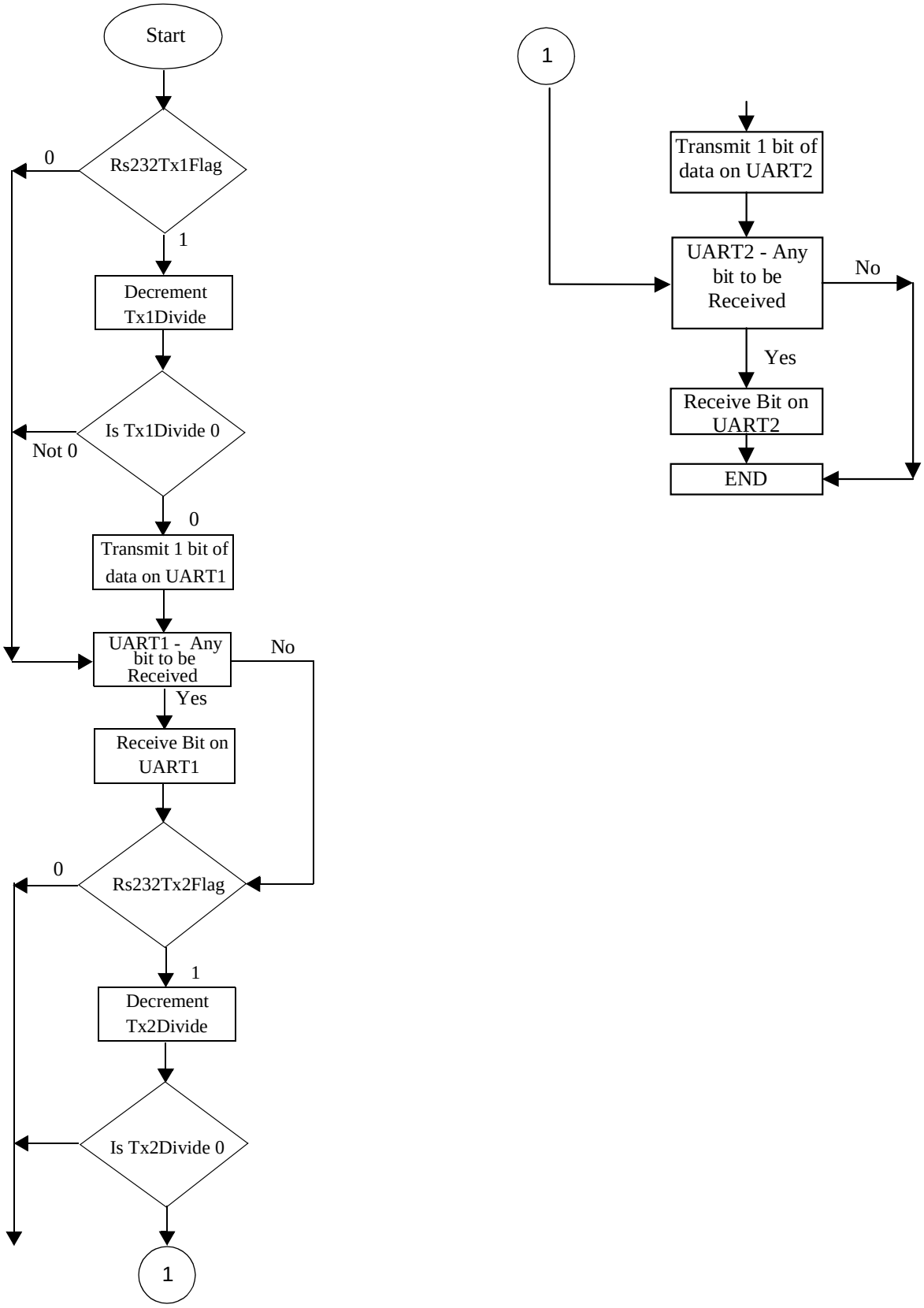

Figure 1-1. Interrupt Service Routine Flowchart

2.0 Source Code Sections

The source code for the UART Virtual Peripheral is divided into four sections:

- Equates Section
- Bank Section
- Initialization Section
- Interrupt Section

When integrated into an application, each section of the source code is inserted at an appropriate location in the main body of the application's source code.

2.1 Equates Section

The equates section provides the values of `UARTDivide` and `UARTStDelay` and the port pin declarations.

The values of the constants are:

```
UARTfs        = 230400
Num           = 4
Int Period    = 217
UARTDivide1 = UARTfs/(UARTbaud1  * Num)
UARTStDelay1 = UARTDivide1 + (UARTDivide1/2)+1
UARTDivide2 = UARTfs/(UARTbaud2  * Num)
UARTStDelay2 = UARTDivide2 + (UARTDivide2/2)+1
```

`Num` is the number of times the UART Virtual Peripheral ISR is called by the multitasker during one rotation. The multitasker rotates interrupt service among 16 slots, and `isrThread1` is called from four of these slots, so `Num` is 4 in this example. In other applications, `Num` might have a different value. For example, if the interrupt frequency were faster or the baud rate were slower, one slot might be sufficient to service the Dual UART Virtual Peripheral ISR.

The pins for sending and receiving data are defined in this section. Port A and Port C are used for the external interface.

The pins are configured as shown below:

```
rs232Rxpin1  equ   ra.2  ;UART1 receive input
rs232Txpin1  equ   ra.3  ;UART1 transmit output
rs232Rxpin2  equ   rc.7  ;UART2 receive input
rs232Txpin2  equ   rc.6  ;UART2 transmit output
```

The baud rates for each of the UARTs are specified by using IFDEF statements. The baud rate is equal to the number that represents it in the commented statement. For example, if `U1B1200` is uncommented, UART1 has a baud rate of 1200 baud. Similarly, if `U2B1920` is uncommented, UART2 is configured for a baud rate of 19200 baud.

2.2 Bank Section

This section describes the use of the banks in the Dual UART Virtual Peripheral. Two banks are used, bank 1 and bank 2.

Different banks for `rs232TxBank` and `MultiplexBank` are defined in bank 1 for clarity, and bank2 contains `rs232RxBank`.

```
       Org    bank1_org


bank1               =       $
rs232TxBank         =       $     ;UART Transmit bank
rs232Tx1high        ds      1     ;High Byte to be transmitted
rs232Tx1low         ds      1     ;Low Byte to be transmitted
rs232Tx1count       ds      1     ;counter to keep track of the bits sent
rs232Tx1divide      ds      1     ;xmit timing counter
rs232Tx1Byte        ds      1     ;store the byte to be sent in this buffer

rs232Tx2high        ds      1     ;High Byte to be transmitted
rs232Tx2low         ds      1     ;Low Byte to be transmitted
rs232Tx2count       ds      1     ;counter to keep track of the bits sent
rs232Tx2divide      ds      1     ;xmit timing counter
rs232Tx2Byte        ds      1     ;store the byte to be sent in this buffer

MultiplexBank       =      $      ;Multipler Bank
isrMultiplex        ds     1      ;Used to jump between the Isr Threads when
                                  ; An Interrupt occurs

       Org    bank2_org

bank2               =      $
rs232RxBank         =      $
rs232Rx1count       ds       1    ;counter to keep track of the number of bits received
rs232Rx1divide      ds       1    ;receive timing counter
rs232Rx1byte        ds       1    ;buffer for incoming byte
rs232byte1          ds       1    ;Used by serial routines
rs232Rx2count       ds       1    ;counter to keep track of the number of bits received
rs232Rx2divide      ds       1    ;receive timing counter
rs232Rx2byte        ds       1    ;buffer for incoming byte
rs232byte2          ds       1    ;used by serial routines
```

2.3 Initialization Section

This section provides initialization for the UART Virtual Peripheral. This has to be included with the initialization of all other ports and registers, prior to entering the main loop of the application.nitialization is required to send the data at the desired baud rate. The value of `UART1divide` specifies the number of times the interrupt has to be serviced before a bit is transmitted. For example, at 9600 baud the value of `UART1divide` is 6, which means that a bit is transmitted once for every six times `isrThread1` is called.

```
_bank rs232TxBank                ;select rs232 bank

mov w,#UARTdivide1               ;load Txdivide with UART baud rate
mov rs232Txdivide1,w

mov w,#UARTdivide2               ;load Txdivide with UART baud rate
mov rs232Txdivide2,w
```

2.4 Interrupt Section

The flow of the interrupt service routine is shown in Figure 1-1.

The ISR returns with a "retiw" value of -217 every 4.32 microseconds at an oscillator frequency of 50 MHz.

```
;*************************************************************************************************
             org    INTERRUPT_ORG              ; First location in program memory.
;*************************************************************************************************
;*************************************************************************************************
;---------------------------------- Interrupt Service Routine -----------------------------
; Note: The interrupt code must always originate at address $0.
; Interrupt Frequency = (Cycle Frequency / -(retiw value))  For example:
; With a retiw value of -217 and an oscillator frequency of 50MHz, this
; code runs every 4.32us.
;*************************************************************************************************
             org     $0
interrupt                                   ;3

;*************************************************************************************************
; Interrupt
; Interrupt Frequency = (Cycle Frequency / -(retiw value))  For example:
; With a retiw value of -217 and an oscillator frequency of 50MHz, this code runs
; every 4.32us.
;*************************************************************************************************

;*************************************************************************************************
;-----------------------------------------VP:VP Multitasker-------------------------------
; Virtual Peripheral Multitasker : up to 16 individual threads, each running at the
; (interrupt rate/16). Change then below:
;Input variable(s): isrMultiplex: variable used to choose threads
;Output variable(s): None,executes the next thread
;Variable(s) affected: isrMultiplex
;Flag(s) affected: None
;Program Cycles: 9 cycles (turbo mode)
;*************************************************************************************************
               _bank          Multiplexbank        ;
               inc            isrMultiplex         ; toggle interrupt rate
               mov            w,isrMultiplex       ;
;*************************************************************************************************
; The code between the tableStart and tableEnd statements MUST be completely within the first
; half of a page. The routines it is jumping to must be in the same page as this table.
;*************************************************************************************************
         tableStart                                ; Start all tables with this macro
               jmp            pc+w                 ;
               jmp            isrThread1           ;
               jmp            isrThread2           ;
               jmp            isrThread3           ;
               jmp            isrThread4           ;
               jmp            isrThread1           ;
               jmp            isrThread5           ;
               jmp            isrThread6           ;
               jmp            isrThread7           ;
               jmp            isrThread1           ;
               jmp            isrThread8           ;
               jmp            isrThread9           ;
               jmp            isrThread10          ;
               jmp            isrThread1           ;
               jmp            isrThread11          ;
```

```
                jmp             isrThread12          ;
                jmp             isrThread13          ;
          tableEnd                                   ; End all tables with this macro.
;*************************************************************************************
;VP: VP Multitasker
; ISR TASKS
;*************************************************************************************
isrThread1                                           ; Serviced at ISR rate/4
;*************************************************************************************
; Virtual Peripheral: Universal Asynchronous Receiver Transmitter (UART) These routines send
; and receive RS232 serial data, and are currently configured (though modifications can be
; made) for the popular "No parity-checking, 8 data bit, 1 stop bit" (N,8,1) data format.
;
; The VP below has 2 UARTS implemented - UART1 & UART2. Both the UARTs can work at independent
; Baud Rates.
;
; RECEIVING: The rs232Rx1flag & rs232Rx2flag are set high whenever a valid byte of data has
; been received and it is the calling routine's responsibility to reset this flag once the
; incoming data has been collected.
;
; TRANSMITTING: The transmit routine requires the data to be inverted and loaded
; (rs232Txhigh+rs232Txlow) register pair (with the inverted 8 data bits stored in
; rs232Txhigh and rs232Txlow bit 7 set high to act as a start bit). Then the number of bits
; ready for transmission (10=1 start + 8 data + 1 stop) must be loaded into the rs232Txcount
; register. As soon as this latter is done, the transmit routine immediately begins sending
; the data. This routine has a varying execution rate and therefore should always be
; placed after any timing-critical virtual peripherals such as timers,
; adcs, pwms, etc.
; Note:The transmit and receive routines are independent and either may be removed for each
;       of the UARTs. The initial "_bank rs232TxBank" & "_bank rs232RxBank" (common)
;       instruction is kept for Transmit & Receive routines.
;       Input variable(s):       rs232Tx1Low (only high bit used), rs232Tx1High, rs232Tx1Count
;                                If rs232Tx1Flag SET, then transmit on UART1
;                                rs232Tx2Low (only high bit used), rs232Tx2High, rs232Tx2Count
;                                If rs232Tx2Flag SET, then transmit on UART2
;       Output variable(s):      rs232Rx1Flag, rs232Rx1byte
;                                rs232Rx2Flag, rs232Rx2byte
;       Variable(s) affected:    rs232Tx1divide, rs232Rx1divide, rs232Rx1count
;                                rs232Tx2divide, rs232Rx2divide, rs232Rx2count
;       Flag(s) affected:        rs232Tx1Flag, rs232Tx2Flag
;                                rs232Rx1Flag, rs232Rx1Flag
;      Program cycles:           22 worst case for Tx, 23 worst case for Rx
;      Variable Length?          Yes.
;*************************************************************************************
UART1
rs232Transmit
             _bank        rs232TxBank         ;2 switch to serial register bank

             sb           rs232Tx1FLag        ;1
             jmp          rs232Receive1       ;1
             decsz        rs232Tx1divide      ;1 only execute the transmit routine
             jmp          rs232Receive1       ;1
             mov          w,#UARTDivide1      ;1 load UART baud rate (50MHz)
             mov          rs232Tx1divide,w    ;1
             test         rs232Tx1count       ;1 are we sending?
             snz                              ;1
             jmp          rs232Receive1       ;1
```

```
:txbit      clc                         ;1 yes, ready stop bit
            rr          rs232Tx1high    ;1 and shift to next bit
            rr          rs232Tx1low     ;1
            dec         rs232Tx1count   ;1 decrement bit counter
            snb         rs232Tx1low.6   ;1 output next bit
            clrb        rs232TxPin1     ;1
            sb          rs232Tx1low.6   ;1
            setb        rs232TxPin1     ;1
            test        rs232Tx1count   ;1 are we sending?
            snz                         ;1
            clrb        rs232Tx1Flag    ;1,22

rs232Receive1
            _bank       rs232RxBank     ;2
            sb          rs232RxPin1     ;1 get current rx bit
            clc                         ;1
            snb         rs232RxPin1     ;1
            stc                         ;1
            test        rs232Rx1count   ;1 currently receiving byte?
            sz                          ;1
            jmp         :rxbit          ;1 if so, jump ahead
            mov         w,#9            ;1 in case start, ready 9 bits
            sc                          ;1 skip ahead if not start bit
            mov         rs232Rx1count,w ;1 it is, so renew bit count
            mov         w,#UARTStDelay1 ;1 ready 1.5 bit periods (50MHz)
            mov         rs232Rx1divide,w ;1
:rxbit      decsz       rs232Rx1civide  ;1 middle of next bit?
            jmp         :rs232RxOut1    ;1
            mov         w,#UARTDivide1  ;1 yes, ready 1 bit period (50MHz)
            mov         rs232Rx1divide,w ;1
            dec         rs232Rx1count   ;1 last bit?
            sz                          ;1 if not
            rr          rs232Rx1byte    ;1 then save bit
            snz                         ;1 if so,
            setb        rs232Rx1Flag    ;1,23 then set flag
:rs232RxOut1


UART2
            _bank       rs232TxBank     ;2 switch to serial register bank
            sb          rs232Tx2flag    ;1
            jmp         rs232Receive2   ;1
            decsz       rs232Tx2divide  ;1 only execute the transmit routine
            jmp         rs232Receive2   ;1
            mov         w,#UARTDivide2  ;1 load UART baud rate (50MHz)
            mov         rs232Tx2divide,w ;1
            test        rs232Tx2count   ;1 are we sending?
            snz                         ;1
            jmp         rs232Receive2   ;1

:txbit      clc                         ;1 yes, ready stop bit
            rr          rs232Tx2high    ;1 and shift to next bit
            rr          rs232Tx2low     ;1
            dec         rs232Tx2count   ;1 decrement bit counter
            snb         rs232Tx2low.6   ;1 output next bit
            clrb        rs232TxPin2     ;1
```

```
             sb          rs232Tx2low.6       ;1
             setb        rs232TxPin2         ;1
             test        rs232Tx2count       ;1 are we sending?
             snz                             ;1
             clrb        rs232Tx2Flag        ;1,22

rs232Receive2
             _bank       rs232RxBank         ;2
             sb          rs232RxPin2         ;1 get current rx bit
             clc                             ;1
             snb         rs232RxPin2         ;1
             stc                             ;1
             test        rs232Rx2count       ;1 currently receiving byte?
             sz                              ;1
             jmp         :rxbit              ;1 if so, jump ahead
             mov         w,#9                ;1 in case start, ready 9 bits
             sc                              ;1 skip ahead if not start bit
             mov         rs232Rx2count,w     ;1 it is, so renew bit count
             mov         w,#UARTStDelay2     ;1 ready 1.5 bit periods (50MHz)
             mov         rs232Rx2divide,w    ;1
:rxbit       decsz       rs232Rx2civide      ;1 middle of next bit?
             jmp         :rs232RxOut2        ;1
             mov         w,#UARTDivide2      ;1 yes, ready 1 bit period (50MHz)
             mov         rs232Rx2divide,w    ;1
             dec         rs232Rx2count       ;1 last bit?
             sz                              ;1 if not
             rr          rs232Rx2byte        ;1 then save bit
             snz                             ;1 if so,
             setb        rs232Rx2Flag        ;1,23 then set flag
:rs232RxOut2

UARTOut

;*************************************************************************************************
;================= PUT YOUR OWN VPs HERE=====================
; Virtual Peripheral:
;
;      Input variable(s):
;      Output variable(s):
;      Variable(s) affected:
;      Flag(s) affected:
;*************************************************************************************************
;-------------------------------------------------------------------------------------------------
             jmp         isrOut      ; 7 cycles until mainline program resumes execution
;-------------------------------------------------------------------------------------------------
isrThread2                           ; Serviced at ISR rate/16
;-------------------------------------------------------------------------------------------------
             jmp         isrOut      ; 7 cycles until mainline program resumes execution
;-------------------------------------------------------------------------------------------------
isrThread3                           ; Serviced at ISR rate/16
;-------------------------------------------------------------------------------------------------
             jmp         isrOut      ; 7 cycles until mainline program resumes execution
;-------------------------------------------------------------------------------------------------
isrThread4                           ; Serviced at ISR rate/16
;-------------------------------------------------------------------------------------------------
             jmp         isrOut      ; 7 cycles until mainline program resumes execution
;-------------------------------------------------------------------------------------------------
```

```
isrThread5                          ; Serviced at ISR rate/16
;-------------------------------------------------------------------------------------------
            jmp         isrOut      ; 7 cycles until mainline program resumes execution
;-------------------------------------------------------------------------------------------
isrThread6                          ; Serviced at ISR rate/16
;-------------------------------------------------------------------------------------------
            jmp         isrOut      ; 7 cycles until mainline program resumes execution
;-------------------------------------------------------------------------------------------
isrThread7                          ; Serviced at ISR rate/16
;-------------------------------------------------------------------------------------------
            jmp         isrOut      ; 7 cycles until mainline program resumes execution
;-------------------------------------------------------------------------------------------
isrThread8                          ; Serviced at ISR rate/16
;-------------------------------------------------------------------------------------------
            jmp         isrOut      ; 7 cycles until mainline program resumes execution
;-------------------------------------------------------------------------------------------
isrThread9                          ; Serviced at ISR rate/16
;-------------------------------------------------------------------------------------------
            jmp         isrOut      ; 7 cycles until mainline program resumes execution
;-------------------------------------------------------------------------------------------
isrThread10                         ; Serviced at ISR rate/16
;-------------------------------------------------------------------------------------------
            jmp         isrOut      ; 7 cycles until mainline program resumes execution
;-------------------------------------------------------------------------------------------
isrThread11                         ; Serviced at ISR rate/16
;-------------------------------------------------------------------------------------------
            jmp         isrOut      ; 7 cycles until mainline program resumes execution
;-------------------------------------------------------------------------------------------
isrThread12                         ; Serviced at ISR rate/16
;-------------------------------------------------------------------------------------------
            jmp         isrOut      ; 7 cycles until mainline program resumes execution
;-------------------------------------------------------------------------------------------
isrThread13                         ; Serviced at ISR rate/16
                                    ; This thread must reload the isrMultiplex register
            _bank  Multiplexbank
            mov    isrMultiplex,#255   ;reload isrMultiplex so isrThread1 will be run on the
                                    ; next interrupt.
            jmp    isrOut           ; 7 cycles until mainline program resumes execution
                                    ; This thread must reload the isrMultiplex register
                                    ; since it is the last one to run in a rotation.
;-------------------------------------------------------------------------------------------
isrOut

;*****************************************************************************
; Set Interrupt Rate
;*****************************************************************************

isr_end
            mov    w,#-intperiod    ;refresh RTCC on return
                                    ;(RTCC = 217 no of instructions executed in the ISR)
            retiw                   ;return from the interrupt

;*****************************************************************************************
; End of the Interrupt Service Routine
;*****************************************************************************************
```

3.0 Baud Rate Generation and Timing

As an example of calculating the parameters which control the timing of the Dual UART Virtual Peripheral, consider transmitting data at 57600 baud with four times oversampling (i.e. a sampling frequency of 230.4 kHz).

Transmission time for 1 bit = 1/57600 seconds

The divide ratio `UARTdivide` for the above example is the sampling rate divided by the baud rate and the number of slots for the Dual UART Virtual Peripheral ISR in the multitasker (i.e. `Num`).

So the formula for `UARTdivide` is:

```
UARTdivide = UARTfs/(UARTbaudrate * Num)
= 230400/(57600 * 4) = 1
```

Therefore, setting `UARTdivide` to 1 results in the desired baud rate. In receive mode, the baud rate is calculated in the same way, except that a constant called `UARTstartdelay` is used to skip over the start bit. This constant is equal to 1.5 times the baud period. Its purpose is to ensure that the bits are sampled near the middle of each pulse. Separate `UARTDivide` and `UARTStDelay` constants are used for each UART (e.g. `UARTStDelay1` is used for UART 1, and `UARTStDelay2` is used for UART 2).

3.1 Circuit Design Procedure

The simplest version of the circuit requires two port pins for transmit and receive. If hardware handshaking is used, additional port lines are required. The hardware interface only requires a driver for converting the voltage level of the signals. The same concept can be used to extend and configure two or more independent UARTs.

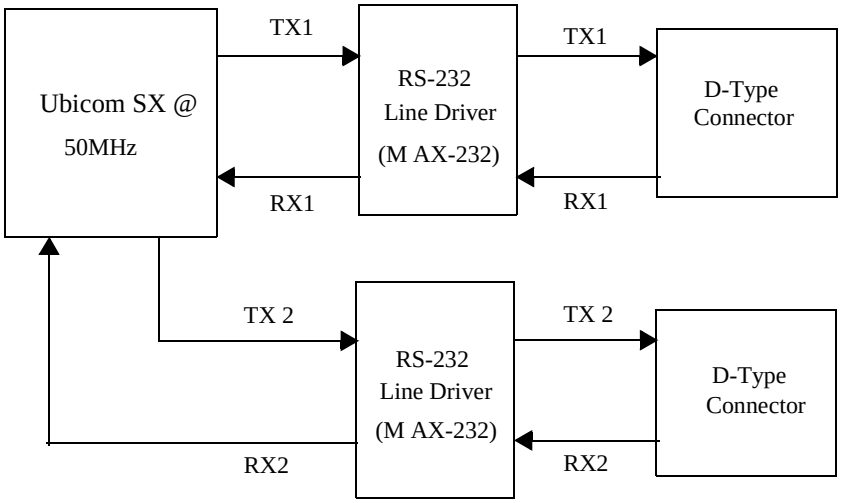

Figure 3-1. Circuit Diagram

4.0 Applications

The program is written for a simple UART without hardware handshaking, but it can be modified to include handshaking.

Because this implementation has two UARTs which can be configured for independent baud rates, it can be used in applications communication with two MCUs or peripherals operating at different baud rates. The Dual UART Virtual Peripheral can be modified by placing the transmit and receive ISRs in different threads, to reduce the service time for each thread.

Lit #: AN39-02

Sales and Tech Support Contact Information

For the latest contact and support information on SX devices, please visit the Ubicom website at www.ubicom.com. The site contains technical literature, local sales contacts, tech support and many other features.

1330 Charleston Road
Mountain View, CA 94043
Contact: sales@ubicom.com
http://www.ubicom.com
Tel.: (650) 210-1500
Fax: (650) 210-8715