SPIC-100X

8 Bit Serial <—> Parallel Transceiver with Host Control

Micro Basic's line of serial transceivers are designed with the hobbyist in mind. They are a welcome tool on every engineer's bench due to their

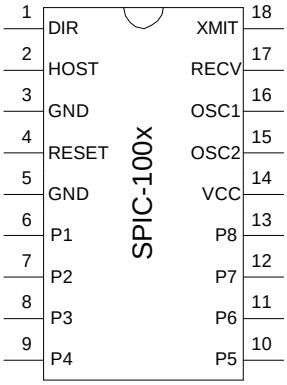
- Low cost
- Ease of use
- Versatility

These serial transceivers have two distinct modes of operation, based on the setting of the **HOST** pin. When set high, the direction of conversion is determined by the setting of the **DIR** pin. When **HOST** is set low, the direction of conversion is determined by commands sent to the SPIC.

The SPIC is based on the Microchip line of PIC microcontrollers. The machine code running on the PIC consists of three real time state machines, and is therefore full-duplex. There are no flow control issues to worry about, nor are any delays necessary.

The bit rate of the serial stream is determined by the oscillator speed, and was optimized for common ceramic resonator values. The following table may be used to determine the oscillator speed needed for common serial speeds:

Bitrate	Oscillator
2400bps	2MHz
4800bps	4MHz
9600bps	8MHz
19.2kbps	16MHz



Devices included in this Data Sheet:

- SPIC-1000
- SPIC-1001
- SPIC-1002

Versatile Serial <—> Parallel Transceiver

- 2 modes of operation
- RS232 standard
- Operating speed: DC to 20MHz clock
  0 to 24kbps
- 8-bit data channel
- No Parity
- 1 Stop Bit
- Multiple oscillator configurations:
  Crystal Oscillator
  Ceramic Resonator
  Clock input (TTL or CMOS)
- External Reset
- True or Inverted RS232 signaling

CMOS technology:

- Low power
- High speed
- Wide operating voltage range
- Wide temperature range
- 25mA sink, 20mA source each pin

© 2000, Micro Basics

Device Variations

The SPIC chips all perform the same function, but accept and send different RS-232 signal polarities:

SPIC-1000 True (Mark = High)
SPIC-1001 Inverted (Mark = Low)
SPIC-1002 Set upon startup

The **SPIC-1000** is meant to be used with a TTL or CMOS to RS-232 level converter, such as the MAX232.
The **SPIC-1001** can connect directly to another serial device through a resister, eliminating the need for an RS-232 level converter.
The **SPIC-1002**, upon reset or power-up, checks the state of the RECV pin and sets its polarity according to that line. This requires that the other RS-232 device be powered up and operating before powering up the SPIC, or a pull-up or pull-down resister on **RECV**. The SPIC-1002 reads the state of the RECV line for the duration of one byte time, including start and stop bits. If the line remains stable for that period of time, it adopts that level as an RS-232 Mark.

All three variations receive and transmit in RS-232, 8 bit, No parity, 1 stop bit.

Modes of operation

Remote

In remote mode the SPIC waits for commands sent via the serial line.
When the ascii character 'I' (Decimal 73, HEX '49') is received the SPIC sets **DIR** low, changes P[1:8] to high-impedance inputs, reads the state of P[1:8], and sends a single byte to represent their state.
When the ascii character 'O' (Decimal 79, HEX '4F') is received the SPIC waits for the next byte. When that byte is received, it is written to P[1:8]. If needed, P[1:8] are set as outputs, and **DIR** is then set high.

This mode is most useful when communicating with an intelligent host, when polled communications is needed, or when software direction control is necessary.

Local

Local mode is chosen by bringing the **HOST** pin high.
In local mode the **DIR** input determines whether the SPIC is converting serial to parallel or parallel to serial. When **DIR** is high the SPIC waits for bytes and writes any byte received to P[1:8]. This mode can receive data at the fastest RS-232 rate, there does not need to be any time between the end of a stop bit and the beginning start bit of the next byte.
When **DIR** is low the SPIC sends a continuous stream of data. Each byte in the stream represents the state of P[1:8] read just before the byte is sent. Since many hosts cannot handle this stream it is most commonly used when communicating with another SPIC set to receive such a stream.
This can be used to transmit the status of 8 bits from one circuit to another using only two wires. At their fastest speed the receiving SPICs outputs are updated 2,400 times per second.

Oscillator Configuration

The SPIC has an on-board low power crystal driver which can drive external crystals and ceramic resonators. The SPIC can also accept an external clock signal (TTL or CMOS).
There are some limitations to the

© 2000, Micro Basics

driver:

- It is set for 4MHz and faster crystals or resonators. Slower crystals or resonators may not run without a series resister, and in some cases will not run at all. It is suggested that slower speeds be implemented with an external clock.
- There are two different SPIC speed ranges. The package markings identify the maximum speed of the SPIC, either 4MHz or 20MHz. While these devices may run faster than rated in a prototyping environment, they are not guaranteed to do so reliably, and the user is cautioned against using them above their ratings in a production unit.
- The SPIC is designed to be used with common oscillator frequencies, generally multiples of 1MHz. This design does not produce an exact bit rate. The bit rate is 0.160% faster than the bit rates shown in the table on the first page. The actual bit rate can be found by the following formula:

$$BitRate \quad (bps) = \frac{F_{OSC}}{832}$$

This is not an issue when the SPIC is communicating with another SPIC. It is not an issue with other RS-232 receivers, which generally allow a bit rate error of greater than 2% before reception errors occur.

Remote/local

© 2000, Micro Basics

Figure 1 is a circuit diagram showing the power, oscillator and reset connections common to all circuits using the SPIC. In cases where a reset button is not necessary, **RESET** may be connected directly to Vcc. A ceramic resonator may be used in place of X1. Ceramic resonators with built-in capacitors can be used to replace X1, C1 and C2. Not shown is a decoupling capacitor, which is placed across Vcc and Vdd.

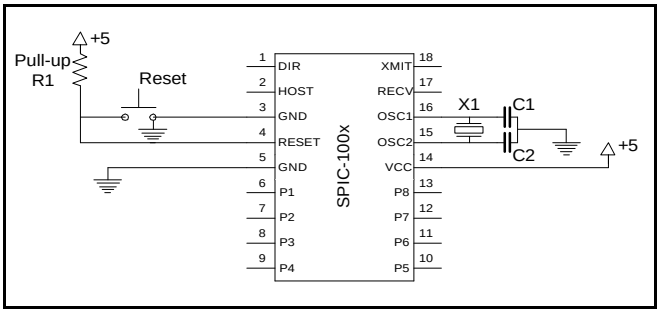


Figure 1

Figure 2 shows the RS-232 connection using a standard MAX232 RS-232 driver/receiver. This method prevents problems associated with static, voltage spikes and other line issues, which are buffered by the MAX232.

This method is only suitable for the SPIC-1000 and SPIC-1002.

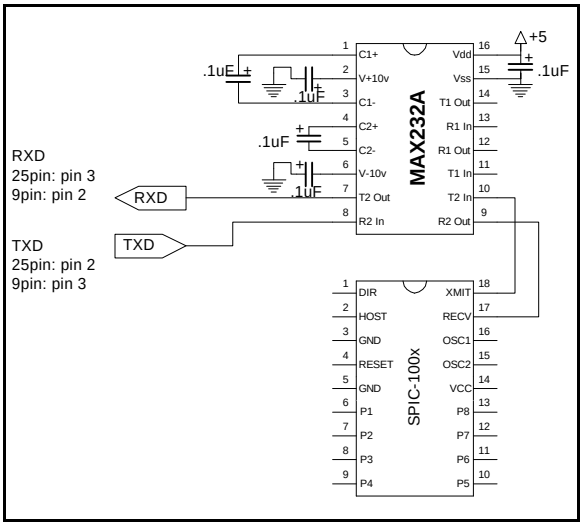


Figure 2

Figure 3 shows the RS-232 connection using a resister. This method is based on two items:

1. The SPIC contains an internal clamping diode on the RECV pin which limits the voltage to 5v on that pin. Coupled with the 22k resister, this clips the RS-232 signal to 0v (Mark) and 5v (Space) levels.

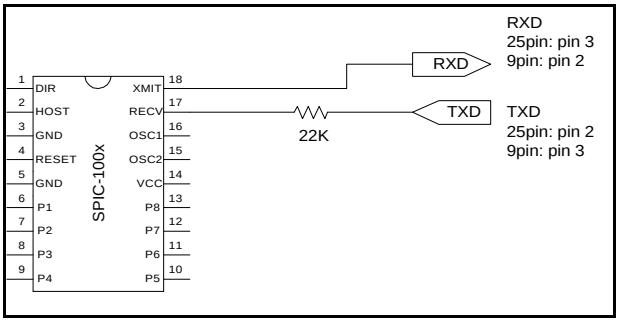


Figure 3

2. While the region of –3v to 3 v is unspecified in the RS-232 specification, most implementations of RS-232 receivers detect 0v as a Mark, and anything above 3v as a space. **XMIT** does not use a level converter, and instead drives the line high (Space) or low (Mark).

This method is only suitable for the SPIC-1001 and SPIC-1002.

© 2000, Micro Basics

Figure 4 shows a complete application of two SPICs. In this circuit both SPICs are configured for local control (**REMOTE** is high). One SPIC is configured as a Parallel to Serial converter, while the other is configured as a Serial to Parallel converter. The LEDs reflect the settings of the switches.

The optional pull-up resisters are only needed when using the SPIC-1002.

While the voltage levels used to send data from one SPIC to the other are not meant for long distances, it is trivial to add either RS-232 or RS-485 converters to separate these SPICs by several thousand meters.

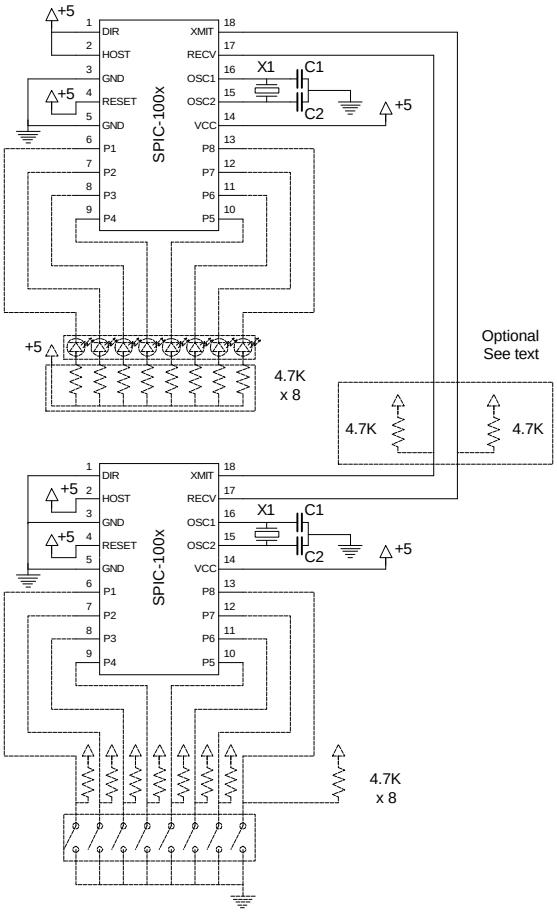


Figure 4

Common Issues

1. Make sure you have a decoupling capacitor across Vcc and Vdd (Gnd).

2. Do not let inputs float: **REMOTE, DIR, RECV, P[1:8]** (when configured as inputs)

3. Do not go over the speed rating of the SPIC.

4. The serial stream output from a SPIC configured as **LOCAL** and Parallel —> Serial is very fast. It can cause buffer overflows on some systems, which may cause system hangs and crashes.

© 2000, Micro Basics

```
; Micro Basics RS-232 Serial<-->Parallel Transceiver
; By: Adam Davis, Spring 2000
; For more information, including an almost complete
; data sheet, please see http://www.ubasics.com/
; This file and the following may change (We reserve
; the right to do that ;-), please check the afore-
; mentioned website for updates periodically.

; Micro Basics and Adam Davis do not provide any
; warrenty or gurantee as to the fitness of this code
; for any particular application or use, and to not
; accept any liability for its use or misuse.

; This file is released into the public domain with
; the following usage restrictions:

; Any item sold with this code (or more than 80% of
; this code) must include (in either electronic or
; paper form) this exact file with no changes, or the
; most recent version of this file to be found on
; www.ubasics.com or officially distributed by Micro
; Basics or Adam Davis unless written permission has
; been granted by Adam Davis or by an executive of
; Micro Basics.  This file must be prominently placed.

; 'Item' shall be defined as, but not limited to, any
; device, record, book, electronic communication, or
; any medium or method which could hold this code in
; any form.  'Sold' shall be taken to mean any
; transaction taking place between two or more
; individuals or businesses in which goods, money, or
; other material/information is mutually exchanged.

; My interpretation is simply:
; Don't sell this without giving your customer the
; 1) knowledge of who created it
; 2) ability to make them
; Don't bundle this with any product without this
; entire document included.

; A large portion of this code came from the book
; "Serial PIC'n", Square One, Roger L. Stevens, without
; which this construction would have taken much longer.
; While not required, I'm acknowleging his contribution.
; I heartily recommend this book as an addition to
; every PIC hobbyist's bookshelf.

; Please note that I like my tabs to be 3 spaces.  The
; comments (and even code) will appear to be spaced
; 'weirdly' when viewed at the normal tab spacing of 8.


      radix   hex

      list p=16c54
;       #DEFINE InvertRS232    ; Comment this out for true RS232
       __CONFIG H'000A'   ; Bit 3: Code Protect (0 for on)
                          ; Bit 2: Watch Dog Timer (1 for on)
                          ; Bits 1 & 0:
                          ; 11 = RC osc
                          ; 10 = HS osc
                          ; 01 = XT osc
                          ; 00 = LP osc
;
c      equ     0
```


© 2000, Micro Basics

```
z       equ     2       ; Carry
dc      equ     1       ; Zero
pd      equ     3       ; Digit Carry
to      equ     4       ; Power Down
;
w       equ     0         ; Working register flag
f       equ     1       ; File register flag
;
;-------------------------------------------------------------------------
indf    equ     00      ; Indirect file register
tmr0    equ     01      ; Timer
pcl     equ     02      ; Program Counter
status  equ     03      ; Status Register
fsr     equ     04      ; Indirect data pointer
porta   equ     05      ; Port A
portb   equ     06      ; Port B
;-------------------------------------------------------------------------
comflg  equ     0008            ; Communications flags
dlyctr  equ     0009            ; Delay Counter
recreg  equ     000A            ; Receive Register
recuse  equ     000B            ; Receive User Register
rloop   equ     000C            ; Receive Loop Counter
rbit    equ     000D            ; Receive Bit Counter
xmtreg  equ     000E            ; Transmit Register
xmtuse  equ     000F            ; Transmit User Register
xloop   equ     0010            ; Transmit Loop Counter
xbit    equ     0011            ; Transmit Bit Counter
;
useflg  equ     0012            ; User Flags (not used?)
myflag  equ     0013               ; My Flags
trisreg equ     0014               ; Copy of Tris B
lochost equ     0015               ; Local/Host register
;
;-------------------------------------------------------------------------
recflg  equ     00      ; Receive Flag
lbflg   equ     01      ; Last Bit Flag
feflg   equ     02      ; Framing Error Flag
ndrflg  equ     03      ; New Data Received Flag
roflg   equ     04      ; Receive Overrun Flag
xmtflg  equ     05      ; Xmitting Flag
xdrflg  equ     06      ; Xmit Data Ready Flag
xcflg   equ     07    ; Xmit Complete Flag

lochospin equ   03      ; Bit for Local/Host Flag
dirpin  equ     02    ; Bit for direction pin

lbi   equ     00       ; Last Byte was Instruction Flag
setup   equ     01    ; Re-Initialize flag
;
outport equ     porta   ; Xmit is Port A:1
outpin  equ     01
inport  equ     porta   ; Receive is Port A:0
inpin   equ     00
;-------------------------------------------------------------------------
         org       0x1ff  ; Reset Vector for 16C54
         goto   start
        org     0x00
;
start   clrf    porta

    ifdef InvertRS232
        bcf     outport,outpin  ; Output marking
    else
        bsf     outport,outpin  ; Output marking
    endif
```

```
        clrf    portb
        movlw   0xFD
          tris  porta           ; Port A all input except bit 1
          movlw   0xFF             ; Set PORTB as inputs
          movwf   trisreg
          tris    portb
        clrf    comflg          ; Clear all com flags
        bsf     comflg,xcflg    ; Set Xmit Completed Flag (MUST DO!)
          bsf       myflag, setup    ; Make sure we go through setup first thing
          bcf       myflag, lbi      ; Clear Last Bit Instruction flag
        goto    frec            ; Start RECV/XMIT/USER cycle (21/19/12 cycles)
;-------------------------------------------------------------------------
;        include taskfull.grp
;====== TASKFULL.GRP ======================================================
;------RECEIVE ROUTINE 21 cycles-----------------------------------------
rdeld   set     rdele           ; Delay rdeld is the same as delay rdele
;
frec    btfss   comflg,recflg   ; Currently Receiveing?  If yes, skip next
        goto    frec2           ; No.  Go see if we have a start bit yet
        decfsz  rloop,f         ; Yes.  Update loop counter.
                                        ; If bit is complete skip next.
        goto    rdelb           ; Bit not complete.  Delay and exit.
        btfsc   comflg,lbflg    ; Last bit flag set? If no, skip next
        goto    frec3           ; Yes.  Go complete byte.
        bsf     status,c        ; No.  Read bit (set carry)
    ifdef InvertRS232
        btfsc   inport,inpin    ; Is b3 = 1? If yes, skip next
    else
        btfss   inport,inpin    ; Is b3 = 1? If yes, skip next
    endif
        bcf     status,c        ; No.  Clear Carry.
        rrf     recreg,f        ; Rotate carry to received byte.
        movlw   04              ; Initialize loop counter
        movwf   rloop
        decfsz  rbit,f          ; Update rbit counter.
                                       ; Skip next if all 8 bits done.
        goto    rdele           ; Not all 8 are done. Delay and exit
        btfss   comflg,ndrflg   ; New data received flag clear?  If yes, skip next
        bsf     comflg,roflg    ; No. Set receive overrun flag.
        bsf     comflg,lbflg    ; Set last bit flag
        movf    recreg,w        ; Move received byte to user receive register
        movwf   recuse
        goto    rdelf           ; Delay and exit
;
frec2
    ifdef InvertRS232
          btfss   inport,inpin    ; Start bit?  If yes, skip next.
    else
          btfsc   inport,inpin    ; Start bit?  If yes, skip next.
    endif
        goto    rdela           ; No.  Delay and exit
        bsf     comflg,recflg   ; Yes.  Set receiving flag
        movlw   08              ; Initialize receive bit counter
        movwf   rbit
        movlw   05              ; Initialize receive loop counter
        movwf   rloop
        goto    rdelc           ; Delay and exit
;
frec3
    ifdef InvertRS232
          btfss   inport,inpin    ; Stop bit?  If no, skip next
    else
          btfsc   inport,inpin    ; Stop bit?  If no, skip next
    endif
```


© 2000, Micro Basics

```
        goto    frec4           ; Yes. Skip next instruction.
        bsf     comflg,feflg    ; Set Framing error flag
frec4   bcf     comflg,recflg   ; Clear receiving flag
        bcf     comflg,lbflg    ; Clear last bit flag
        bsf     comflg,ndrflg   ; Set new data received flag
        goto    rdeld           ; Delay and Exit.

rdelb   nop

                        ; This is the setup/initialization routine.
                        ; It is run inbetween bit readings and between bytes.
                        ; It is possible for the operate routine (after xmit routine)
                        ; to be run before this setup.  If so, and if any setup
                        ; changes needed to be made beforehand it could mess
                        ; things up.  This can only occur when the received stop
                        ; bit is less than 3/4 of one bit time.

rdela     btfss  myflag, setup       ; Setup flag? If yes, skip next
          goto   dlysetup            ; Delay and exit
          btfsc    porta, lochospin  ; Local or host? If local(0), skip next
          goto     host                 ; Host(1). Goto host setup
          bcf          myflag, lochospin  ; Clear the local/host flag
          movlw    00FD                 ; Set the dir pin as input
          tris     porta
          btfsc    porta, dirpin     ; Direction in(0) or out(1)? If in, skip next
          goto     localout             ; Out, goto local out setup
          movlw    00FF                 ; In. Set portB as inputs.
          tris     portb
          movwf    trisreg           ; Copy tris B settings to trisreg
          bcf          myflag, setup     ; Clear the setup flag
SupE    goto     fxmt  ; 2 1         ; Goto transmit routine

host
            movlw      00F9                 ; Set direction pin as output
            tris       porta
            bcf    porta, dirpin     ; Clear direction pin, signifying input
            movlw      00FF                 ; Set portb as inputs (safest route)
            tris       portb
            movwf      trisreg           ; Copy portb tris to trisreg
            bcf    myflag, setup     ; Clear the setup flag
            nop                          ; Delay
            goto       fxmt                 ; Goto transmit routine

dlysetup                                 ; Various delays used in both the
          nop             ; 12       ; receive and setup routines
          nop             ; 11
          nop             ; 10
rdelc   nop              ; 9
rda2    goto    rdele  ; 8 7
rdele   goto    rda3      ; 6 5
rda3    goto    rda4      ; 4 3
rda4    goto    rdelf  ; 2 1

localout
            movlw      0000                 ; Set portB as output
            tris       portb
            movwf      trisreg           ; Copy portb tris to trisreg
            bcf    myflag, setup     ; Clear setup flag
            nop                          ; Delay (transmit routine next

rdelf

;----TRANSMIT ROUTINE 19 cycles-----------------------------------------------------
fxmt    btfss   comflg,xmtflg   ; Transmitting? If yes, skip next
        goto    fxmt2           ; No. Go see if we need to be.
```

```
        decfsz  xloop,f         ; Yes.  Update loop counter.
                                        ; If bit completed skip next
        goto    xdlyb           ; Bit not completed.  Delay and exit
        movlw   04              ; Initialize loop counter
        movwf   xloop
        decfsz  xbit,f          ; Update bit counter.
                                        ; Skip next if all bits completed.
        goto    fxmt4           ; Not all bits done. Go send next bit.
        bcf     comflg,xmtflg   ; All bits sent. Clear xmit flag.
        bsf     comflg,xcflg    ; Set xmit complete flag
        goto    xdlyc           ; Delay and exit

;
fxmt2   btfss   comflg,xdrflg   ; xmit data ready? If yes, skip next
        goto    xdlya           ; No. Delay and exit.
        bsf     comflg,xmtflg   ; Yes. Set xmiting flag.
        movlw   0a              ; Initialize bit counter.
        movwf   xbit
        movlw   04              ; Initialize loop counter.
        movwf   xloop
        movf    xmtuse,w        ; Move user data to xmit register.
        movwf   xmtreg
        bcf     comflg,xdrflg   ; Clear xmit data ready flag.
        goto    fxd1            ; Cycle equalizer
fxd1
    ifdef InvertRS232
          bsf     outport,outpin  ; Start bit.
    else
          bcf     outport,outpin  ; Start bit.
    endif
          nop
          nop
          nop
          nop
          nop
          nop
          nop
        goto    fxmt8           ; Delay and exit.
;
;
fxmt4   bsf     status,c       ; Set Carry
        rrf     xmtreg,f        ; Move data bit to Carry
;
fxmt5
    ifdef InvertRS232
          btfss   status,c       ; Carry (data bit) = 0 ? If yes, skip next
    else
          btfsc   status,c       ; Carry (data bit) = 0 ? If yes, skip next
    endif
        goto    fxmt6           ; No. data bit = 1. Set Pin a:1
        bcf     outport,outpin  ;Yes. Data bit =0. Clear port a1
        goto    fxmt7           ; Delay & exit
fxmt6   bsf     outport,outpin  ; set port A:1
        nop                     ; Cycle Equalizer
fxmt7   nop
          nop
          nop
          nop
          nop
          nop
          nop
          goto    fxmt8           ; Delay & exit
;
;
```


© 2000, Micro Basics

```
xdlyc   goto    fxd4      ; 6 5 ; Various delays for xmit routine
fxd4    goto    fxd5      ; 4 3
fxd5    nop
         nop
         nop
         nop
         nop
         nop
         nop
         goto    fxmt8    ; 2 1

match1  movlw      00FF              ; Set Tris B as inputs
           bsf     porta, dirpin  ; Dirpin = high (input)
           goto       main1             ; Go back to main operate routine

nomatch nop
           goto       main2             ; Delay and exit

DoHostOut
           btfss      myflag, lbi    ; Last byte instruction? If yes, skip next
           goto       operF             ; Delay and exit
           btfss      trisreg, 0     ; Tris B as input? If yes, skip next
           goto       operA             ; No. Delay and exit
           btfss      comflg, xcflg  ; Yes.  Is transmitter ready?  If yes, skip next.
           goto       operE             ; No. Delay and exit.
           movf       portb, w          ; Copy Port B to xmtuse
           movwf      xmtuse
           bcf     comflg, xcflg  ; Clear Xmit completed flag
           bsf     comflg, xdrflg ; Set Xmit data ready flag
           bcf     myflag, lbi    ; Clear last byte instruction flag
           goto       main2             ; Go back to main operate routine

DoHostIn   btfsc      trisreg,0      ; Is port b set as output? If yes, skip next.
           goto       operA             ; No. Delay and exit
           movf       recuse, w      ; Move byte from receive register to port b
           movwf      portb
           bcf     porta, dirpin  ; Yes. Set the direction pin low to show output.
           bcf     myflag, lbi    ; Clear last byte instruction flag.
           nop
           nop
           nop
           goto       main2             ; Go back to main operate routine.

dolocal btfsc      porta, dirpin  ; Is the direction pin low? If yes, skip next.
           goto       localin        ; No. Go to input routine.
           movlw      00FF              ; Set port b as input
           tris       portb
           movwf      trisreg        ; Copy port b tris to trisreg
           btfss      comflg, xcflg  ; Is the transmitter free? If yes, skip next.
           goto       operC             ; No.  Delay and exit
           movf       portb,w        ; Yes.  Move port B to xmit register.
           movwf      xmtuse
           bcf     comflg, xcflg  ; Clear xmit complete flag
           bsf     comflg, xdrflg ; Set xmit-ready flag
           goto       operD             ; Delay and exit

localin movlw      0000              ; Set port b as outputs
           tris       portb
           movwf      trisreg        ; Copy port b tris to trisreg
           btfss      comflg, ndrflg ; New data received? If yes, skip next.
           goto       operE             ; No. Delay and exit
           movf       recuse, w      ; Copy received data to port b.
           movwf      portb
           bcf     comflg, ndrflg ; Clear the new data received flag.
           goto       operD             ; Delay and exit
```

```
operF      nop ; Various delay points for the operate routine
           nop
operA      nop
operC      nop
operE      nop
operB      nop
           nop
           nop
operD      nop
           goto   fxmt8

xdlyb   nop               ; 20
xdlya   btfss  porta, lochospin      ; Skip next if HOST
          goto  dolocal              ; Locah/Host pin is low (local control)
          btfss comflg, ndrflg       ; Skip next if there is a character waiting
          goto  DoHostOut            ; No character waiting, do nothing.
          btfsc myflag, lbi          ; Skip next if the last character was NOT '1' or '2'
          goto  DoHostIn                ; The last character tells me to perform a hosted
function
          movlw  '1'                    ; Put '1' into W for a comparison test
          xorwf recuse, w            ; If recuse matches '1' then W will be 0.
          btfsc status, z            ; Skip next if W is not zero
          goto  match1               ; W was zero, set up PORTB, etc
          movlw  '2'                    ; W was not zero, put '2' into W for comparison
          xorwf recuse, w            ; If recuse matches '2' then W will be zero now
          btfss status, z            ; Skip next if W is zero
          goto  nomatch              ; W was not zero, no match, we'll exit.
          movlw  0000                   ; Set W to 0000, which will go into trisB and make
it output
main1     tris  portb                   ; Set port b to all output
          movwf trisreg              ; Copy setting to trisreg for directional use later
          bsf      myflag, lbi          ; Set the 'Last Byte was Instruction' flag
main2     bcf      comflg, ndrflg       ; Reset the receiver
;
fxmt8
;
;-------------------------------------------------------------------------------
;dloop  decfsz  dlyctr,f
;        goto    dloop
;        retlw   0
;-------------------------------------------------------------------------------

;
;------USER ROUTINE 12 cycles-------------------------------------------------
echo       movf       porta, w          ; Copy port A to w.
           andlw      0008              ; Mask w so that pin a:3 is left alone, all others
are cleared
           xorwf      lochost,w      ; Test lochost register against w.
                                        ; If they are the same, w will be zero
           btfsc      status, z      ; Are they the same? If yes, skip next.
           bsf    myflag, setup  ; Set setup flag
uda3    goto    frec            ;Last line of user

dloop  decfsz  dlyctr,f            ;3n+5 (includes call statement and return)
        goto    dloop
        retlw   0
;-------------------------------------------------------------------------
       end
;-------------------------------------------------------------------------
```


© 2000, Micro Basics

Information contained in this publication regarding device applications and the like is intended for suggestion only and may be superseded by updates. No representation or warranty is given and no liability is assumed by Micro Basics with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Micro Basics' products as critical components in life support systems is not authorized except with express written approval by Micro Basics. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights. The Micro Basics logo and name are trademarks of Micro Basics in the U.S.A. and other countries. All rights reserved. All other trademarks mentioned herein are the property of their respective companies.