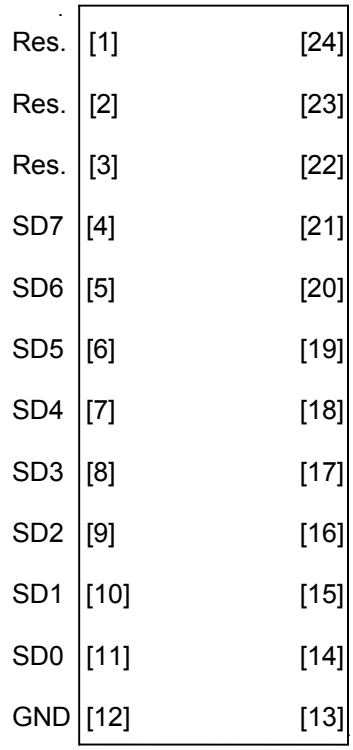
USB245



Pinout Information

Res.	Reserved, Unconnected. Do not use.
SD(0..7)	Data Bus. Driven when READ# is asserted. Data inputs when WRITE is asserted.
RCCLK	Output of the onboard RC Timer.
RXF#	Active low output. When asserted, receive data is available, and can be read from FIFO.
RESET#	Active Low output. Remains low for 140msec after power-up, then goes high.
READ#	Active Low input. When low, the device drives the current receive FIFO data onto SD0-SD7.
WRITE	Input that Clocks Data at SD0-SD7 into transmit FIFO on a High-to-Low transition.
TXE#	Active Low Output. When asserted, signifies transmit FIFO space is available.
DP	USB Plus Data Signal.
DM	USB Minus Data Signal.
USB_VCC	Input to onboard 5volt power filter.
FILTER_VCC	Output of onboard 5volt power filter.
VCC	5 Volt power input.
GND	Ground.

- USB ver1.1 compatible device with a 384 byte transmit FIFO and 128 Byte receive FIFO.
- Works with FTDI's Virtual Com Port (VCP) and D2XX Drivers for Microsoft Windows.
- Small Size: Surface Mount Technology allows the device to fit in standard 24 pin DIP socket.
- Preprogrammed 93C46 EEPROM with unique USB device information.
- Easy to use 8-bit microcontroller-microprocessor interface with FIFO status lines

The USB245 is small, easy to use device with all necessary timing circuits, and power filtering built in.

Each contains a preprogrammed EEPROM which holds the device and vendor ID codes.

The 8-bit data bus easily connects to processor data buses. The two control bus lines (READ# and WRITE) direct data flow from the device, while two status lines (TXE# and RXF#) give information about the state of the internal transmit and receive FIFOs. The status lines can also be used as interrupt sources for the processor.

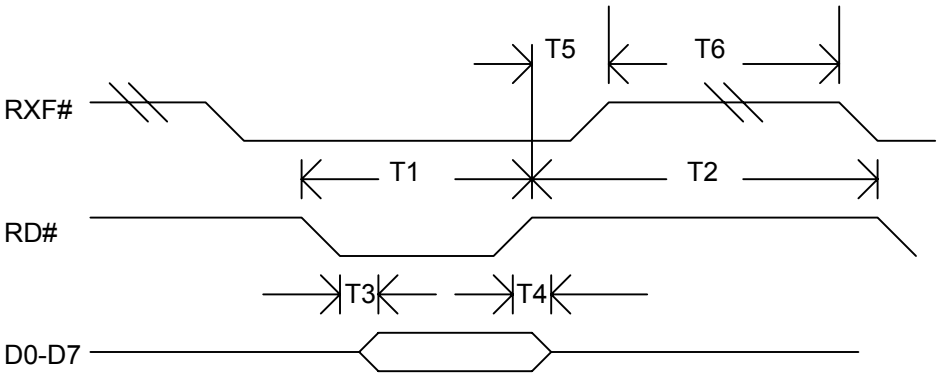
The onboard RESET# line is routed externally and can be used to drive other circuits.

The device can be wired to derive and source power from the USB bus, or alternatively have power sourced from the processor interface.

www.xylotex.com

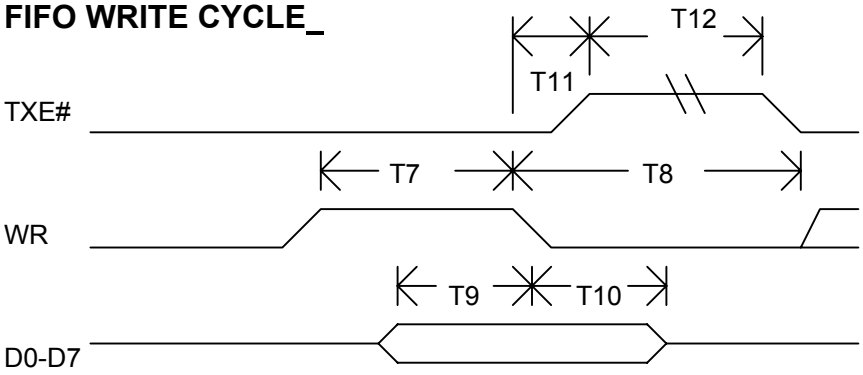
Timing Information

FIFO READ CYCLE



		Min	Max	Unit
T1	RD Active Pulse Width	50		ns
T2	RD to RD Pre-Charge Time	50		ns
T3	RD Active to Valid Data		30	ns
T4	Valid Data Hold from RD inactive	10		ns
T5	RD Inactive to RXF#	5	25	ns
T6	RXF# Inactive after RD cycle	80		ns

FIFO WRITE CYCLE_



T7	WR Active Pulse Width	50		ns
T8	WR to WR Pre-Charge Time	50		ns
T9	Data Setup Time Before WR Inactive		20	ns
T10	Data Hold Time from WR Inactive	10		ns
T11	WR inactive to TXE#	5	25	ns
T12	TXE# inactive after RD cycle	80		ns

Device Function

A processor can use the status lines TXE# and RXF# in either a polled or interrupt manner. The TXE# line will go active LOW indicating that there is room available in the 384 bytes transmit buffer for the processor to place another byte for transmission over the USB bus. The RXF# line will go active LOW when there is at least one byte available in the 128 byte FIFO for the processor to read.

To write data into the transmit FIFO, the following procedure should be used: With the WRITE line LOW, the processor places the data on lines D0-D7. The processor then strobes the WRITE line by bringing it HIGH for a minimum of 50 ns, and then completes the cycle by bringing the WRITE line back LOW.

To read data from the receive FIFO, the processor drives the RD# line LOW. After a maximum time of 30 ns, valid data will be driven on lines D0-D7. The processor reads the data and then drives the RD# line back inactive HIGH.

For USB powered controls, USB power would be fed into Pin 22. The filtered output provided at Pin 23 would then be tied to Pin 24 (Vcc). This power (+5volts) can be used to power other circuits in the design, up to the maximum current draw per USB device. For USB specifications see: www.usb.org/developers.

For System powered controls, +5 volts can be directly fed into Pin 24. Pins 22-23 are left unconnected. Alternatively system power can be fed into Pin 22. Pin 23 would then be tied to Pin 24 for extra system power filtering. In this case USB power is never connected to the device.

USB power and alternate System power sources should not be tied together, except for common grounding (GND).

The USB245 is a 'slave' or 'peripheral' device to a 'master' or 'host' USB controller such as a PC running Windows 98/2000. The host must have appropriate drivers running to access the USB245. Drivers are available for Windows and Mac Systems.

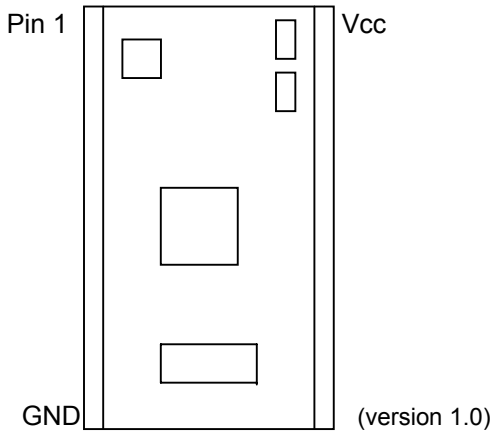
Addition Information

USB Cable color codes: Red→ 5 Volts, Black → GND, Green → DP, White → DM

Schematic Usage

See appendix for a simple example using the USB245 with the Ubicom SX-28.

Pin Orientation



(version 1.0)

Software Drivers Supplied & Supported by: Future Technology Devices International Ltd. (FTDI).
http://www.ftdichip.com

The Following is a sample ‘C’ that reads a character from the USB and writes it back out.

```
#include <c:\sxc\include\dev\SX28AC.H>
#include <c:\sxc\include\port.h>

#define TXEn            PORTA.3  // INPUT
#define WRITE           PORTA.2  // OUTPUT
#define READn           PORTA.1  // OUTPUT
#define RXFn            PORTA.0  // INPUT


#pragma FUSE = FOSC_HS2 & TURBO & WDTD
#pragma FUSEX = EXTEND

unsigned char byte;
/*-------------------------------------------------------------------------*/
/*                                                                         */
/*-------------------------------------------------------------------------*/
void send_byte()
{
DDR(PORTC,0x00);  // make PORTC output
WRITE=1;
NOP();
PORTC=byte;
NOP();
WRITE=0;
DDR(PORTC,0xFF);  // now make PORTC input
}
/*-------------------------------------------------------------------------*/
/*                                                                         */
/*-------------------------------------------------------------------------*/
void read_byte()
{
READn=0;
NOP();
byte=PORTC;
NOP();
READn=1;
}
/*-------------------------------------------------------------------------*/
/*                                                                         */
/*-------------------------------------------------------------------------*/
void main()
{
OPTION(RTW|RTE_D|PSA);  // start with RTC int disabled

DDR(PORTC,0xFF);        // USB Data Port
DDR(PORTB,0xFF);        // all inputs
DDR(PORTA,0xF9);        // USB Status lines

WRITE=0;                // Initialize control
READn=1;                // Outputs Here

while (1)               // Loop forever
      {if (RXFn == 0)   // Is there a character available
          {read_byte(); // Yes, so get it
           send_byte(); // and send it back out
          }
      }
}
```

Setup and running.

Place the USB device in an empty breadboard, noting proper pin orientation. If you are using the USB adapter cable, while it is NOT plugged into a master or PC, plug the proper end into the breadboard next to the USB245 so that the pin with the Red wire connects to USB245 pin 22, and the pin with the black wire connects to pin 19. Place a jumper wire between pins 22 & 23 of the USB245 on the breadboard. At this point only the top-right 6 pins of the USB245 will be attached.

With nothing else attached to the module, except as noted above, now plug the other end of the cable into the PC. It should recognize a new USB device a start searching for drivers. When it

asks for the drivers, put the floppy disk sent with the USB245 into the computer's floppy drive and tell the setup routine which drive you placed it into (usually A:).

After it has completed setup, you can find the COM port the USB245 has been assigned, by clicking on: **Start ➔Settings➔Control Panel**, then double click the **System** Icon ➔ **Device Manager ➔ Ports (COM &LPT)**. Here you should see: **USB Serial Port (COMn)**, where n is the COM port assigned.

Now that it has been found, unplug the USB cable from the PC. **Never** unplug the cable from the USB245 end, while the other end is plugged into a PC or USB master..

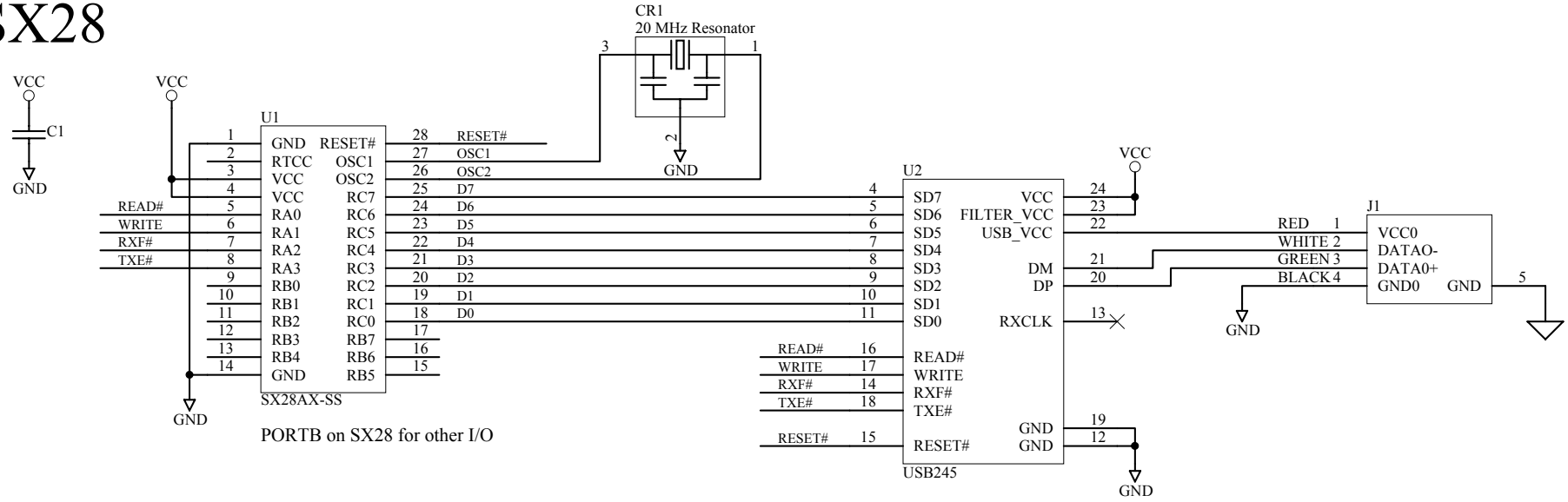
Wire the 8 data lines and 4 status lines to your preprogrammed microcontroller. You can use the 5V supply from pin 24, and the GND from pin 12 for you microcontroller. Do NOT tie another 5V source to the USB 5V source. RESET# is also available if needed.

Plug the USB cable back into the PC. The PC should now search for and find the proper driver software. The PC may 'freeze' for about 25 seconds while the software is loaded. The chipset & driver manufacturer explains that this is normal.

If the example program listed above was downloaded to your embedded processor, you should be able to open a COM program like PCPlus, choose the correct COM Port, and start typing in the data entry area and see you characters show up on the screen. If they show up twice, go to half-duplex mode.

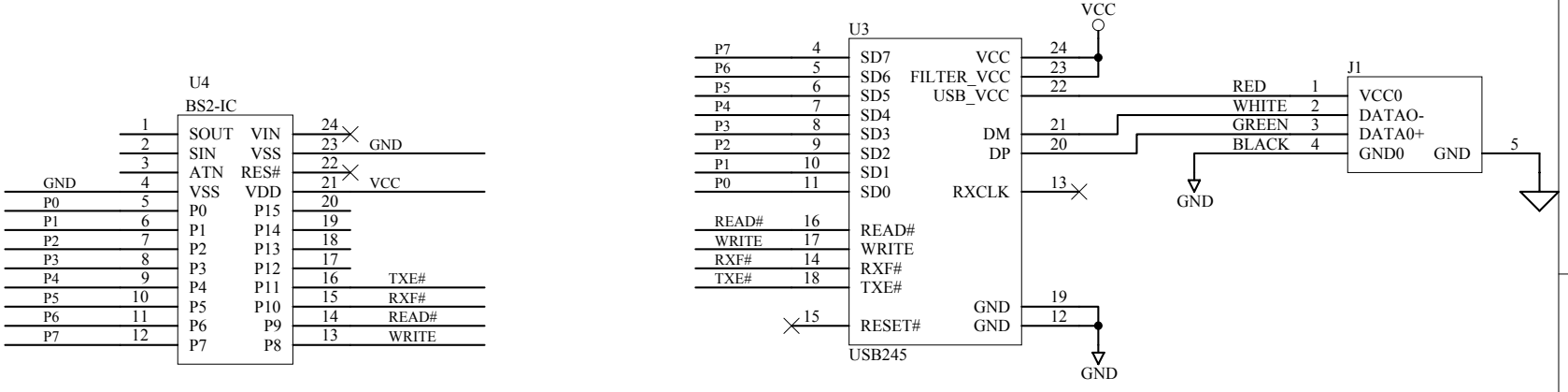
```
A very simple Basic program

USB   VAR  BYTE          REM A Place to store the character
                         REM I/O 0-7 used for DATA
LOW  8                   REM I/O 8  used for WRITE SIGNAL, Start LOW
HIGH 9                   REM I/O 9  used for READ SIGNAL, Start HIGH
                         REM I/O 10 used for RXF# input
                         REM TXE IS NOT CHECKED

10 IF IN10 = 0 THEN      REM IF RXF is LOW, there is a char available
      LOW 9              REM make the READ line active LOW
      USB=INL            REM read the character
      HIGH 9             REM deactivate the  READ line
      DIRL=%11111111     REM set PORT for OUTPUT
      HIGH 8             REM strobe the WRITE line HIGH
      OUTL=USB           REM output the character
      LOW 8              REM bring the strobe back LOW LOW
      DIRL=%00000000     REM set PORT for INPUT
  END
GOTO 10
```

SX28



PORTB on SX28 for other I/O

BS2-IC



Title	USB245 Demo Schematic	
Size: A	Number	Revision
Date: 5-Dec-2002	Sheet of	
File: C:\Designs\stamps.ddb	Drawn By:	