

Re: PIC, Keyboard, and USART

Source: <http://sci.tech-archive.net/Archive/sci.electronics.basics/2006-03/msg01051.html>

- *From:* "Abstract Dissonance" <Abstract.Dissonance@xxxxxxxxxxxx>
 - *Date:* Tue, 28 Mar 2006 23:45:22 -0600
-

"Jan Wagner" <nospam@xxxxxxxxxxxx> wrote in message
[news:QigWf.280\\$qF5.5@xxxxxxxxxxxxxxxxxxxx](news:QigWf.280$qF5.5@xxxxxxxxxxxxxxxxxxxx)

Abstract Dissonance wrote:

I was thinking that all I would have to do is switch between master and slave mode for tranmission and reception.

Basically yes, but, with the additional protocol thingy in between to tell the keyboard that it should start clocking data in and not out.

Not sure if this would work though or not... I'd still have to setup the clock speed properly for tranmission but I figure that it shouldn't be too hard to do cause they are two disjoint parts. The main problem is when worrying about collisions.

The way I've seen this and used the keyboard is that the keyboard actually is always master and provides the clock i.e. tells the PC when it can start sending a command byte. But I'm not an AT keyboard specifications guru, maybe the specs provide for a little-known slave mode also.

I'm not sure. My foggy recollection from a few web sites is that when transmitting to the keyboard you gotta hold the clock line low for a certain amount of time and the keyboard will know then that you are going to transmit something and then will wait. I'm not sure if I'm even close but thats what I remember at this point.

You can find some valuable background info at

<http://www.beyondlogic.org/keyboard/keybrd.htm#1>

Re: PIC, Keyboard, and USART

Yeah, I'm still trying to read up on all this stuff. Its a good site but trying to make some sense of the datasheet for the pic. It gives different methods to do the same thing and I'm confused on which is right.

If you only want to receive, and your uC does support full 11 bits UART ("Enhanced USART" sounds like it might, but then maybe that's just marketing bull...) then this should be very easy to do.

Well, I'm not sure. It says 9-bits but not sure if that includes the start and stop bit ;/

I had a quick glance at the PIC18F2455 datasheet (I like to spot PIC flaws ;-)), and searched the PDF for "parity". Not many search results.

In section 20.2 EUSART Asynchronous Mode the datasheet says that "Parity is not supported by the hardware but can be implemented in software and stored as the 9th data bit.". So the USART might not be that "Enhanced" after all if, it lacks such basic features as parity bit. But it's up to you to check the PIC18 family users guide etc in more depth to see if this is actually the full truth or not (considering it is a PIC, it'd be hardly surprising if the parity feature was really missing ;-)

Anyway, from that one datasheet sentence I'd guess the Enhanced USART does only start + stop bit and 7..9 bits of data. So if you want to /receive/, enable 9 bit mode, and ignore the 9th bit (since at only 9600 baud the chance for errors must be really small :)) For transmitting something, do the parity calculation in software.

yeah, it does start, stop and 8 to 9 data bits but treats the 9th bit as data and not something special like parity. I'm not sure if I need to do parity at this point since I wouldn't even know how to request transmission of data. I'm still trying to figure out how to use the USART for my pic. The docs are pretty technical but I get a little lost in know exactly what to do. They give the info on all the bits and a small description but hell if I know how to use them properly(like in what order and stuff). They do give some example code and I think I'm making it harder than what it is.

I figure that I just have to flip a few bits to set the right mode and put

Re: PIC, Keyboard, and USART

Re: PIC, Keyboard, and USART

some code on an interrupt to handle when I receive the data and hopefully it should work.

Just add the pull-up resistors, data line to UART RX, clock input not strictly necessary – until you want to transmit.

? Is that for asynch mode? I'd need to use the clock in slave mode for the getting data? (since I don't know its data rate). I could use asynch mode and auto detect the clock but I'm not sure how reliable this is ;/ and I think there has to be a calibration test or something and I have no idea how to get that done.

Yup, asynch, no need for the external clock when receiving because basically the keyboard is supposed to be sending at 9600baud (IIRC).

what if the clock is a bit off? Not sure if I want to do asynch yet since I have the clock available. Seems like one less problem to worry about if something doesn't work.

Your PIC UART will (or, /should/ ;–) sync up at the start bit automatically, provided you have set the UART to a fixed 9600 baud and don't use any kind of unreliable "automatic baud rate detection" features.

With UART set to 9600 baud the actual data rate or delay between bytes is irrelevant, as long as bit rate of one individual byte is those 9600 baud – and the PC keyboard takes care of that already.

and if its not? ;/ lol. Damn I wish I would have kept all those keyboards that I ruined by spilling crap on them... Could have looked the what chips they were using and got the data sheet on them ;/

regards,
– Jan

Thanks again,
Jon