

Re: PIC, Keyboard, and USART

Source: <http://sci.tech-archive.net/Archive/sci.electronics.basics/2006-03/msg01006.html>

- *From:* "Abstract Dissonance" <Abstract.Dissonance@xxxxxxxxxxxx>
 - *Date:* Mon, 27 Mar 2006 22:12:57 -0600
-

"Jan Wagner" <nospam@xxxxxxxxxxxx> wrote in message
[news:e05sfv\\$2u0\\$1@xxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:news:e05sfv$2u0$1@xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Abstract Dissonance wrote:

I'm going to try to program a simple keyboard translator using the PIC2455 which has a built in EUSART. At this point I just want to get the keyboard codes into the pic and I just want to run this by you guys to make sure I'm on the right track.

1. Setup hardware. Basicly just have to connect the Data and Clock lines to the EUSART pins on the pic
2. Enable the USART on the pic for Asynchronous mode and enable auto baud rate dection(to make it easy).
3. Read the RCREG in the interrupt and do whatever with it(like make an LED blink on certain codes).

Is that basicly all I have to do? Do I need to use pullups or anything when connecting the lines or even use transistors to source/sink more current?

(I'm running the keyboard off a 5V supply that can source up to .5mA or so)

Apart from the "writing bugs" others already commented on, doing this with UART could be a real nightmare. Ok, 9600 baud and 8 databit, 1 odd parity, 1 start and 1 stop bit, and separate clock input, it is basically easy (if your microcontroller supports that many bits!! 11 of them). But then there is also the bidirectional communication needed especially if you want to configure the keyboard to something sensible by sending commands to it. Basicly you'd have to switch off the UART and put the pins into output mode for the handshakes etc, then switch back to UART again to receive or transmit. For both receive /and/ transmit, you'd have to use the clock provided by the keyboard, i.e. synchronous UART. It's a real mess.

Re: PIC, Keyboard, and USART

yeah... after looking at slave mode I saw this. Difference between master and slave is who supplies the clock.

I was thinking that all I would have to do is switch between master and slave mode for transmission and reception. Not sure if this would work though or not... I'd still have to setup the clock speed properly for transmission 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.

You can find some valuable background info at
<http://www.beyondlogic.org/keyboard/keybrd.htm#1>

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 ;/

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.

regards,
– Jan

Thanks,
Jon

Re: PIC, Keyboard, and USART