

Re: Interrupt driven UART

Source: <http://coding.derkeiler.com/Archive/General/comp.arch.embedded/2006-10/msg00777.html>

- *From:* "goister@xxxxxxxx" <goister@xxxxxxxx>
 - *Date:* 12 Oct 2006 02:52:01 -0700
-

CBFalconer wrote:

Ico wrote:

goister@xxxxxxxx <goister@xxxxxxxx> wrote:

I'm working with a Toshiba TMP91 series MCU that doesn't seem to have any UART control/status bits to check for empty data register, rx/tx ready, etc, but does have interrupt vectors for serial tx and rx, which is why I think I have to use interrupt driven UART rather than polling it.

I have functions that expect to receive and send a single byte by calling receivebyte and sendbyte functions. However, since it's interrupt based, receivebyte seems to be pretty redundant. What I have now is a UART receive ISR that first does error checking, then copies the rx buffer to a global rx byte variable, and setting a global rx_ready flag to 1. Then my receivebyte function does nothing until the rx ready flag is set, after which it just clears it. Does this make sense?

This might work, but imagine what would happen if a byte is received on the uart while your code happens to be doing something else then calling the uart_receive function ? Instead of having only one 'global rx variable', consider using a ringbuffer aka 'circular buffer' aka fifo for this. The RX interrupt stores incoming bytes into the buffer and updates the head and tail pointers, while your uart_receive function reads one byte from the buffer, or – if the buffer is empty – waits until new data comes available.

Re: Interrupt driven UART

The OP will also need to implement some sort of critical section in either case. I imagine this will be most easily implemented by controlling the interrupt enable.

Lacking the data ready signal he will not be able to handle a data overrun during buffer emptying. Lack of those signals sounds ridiculous, maybe he should look again in his data sheets.

--

The receiving buffer is double buffered. According to the datasheet, when the last received bit is stored in buffer1, the stored data is transferred to buffer2(which is CPU accessible), and this causes the rx interrupt to be generated, and I'm using this rx interrupt to call my receive isr. Also, if buffer2 is not read before all bits of the next data are received by buffer1 and overrun error occurs and the overrun status bit is set. buffer1 will then be lost, but buffer2 will be preserved. Seems to me like buffer1 is just a temp buffer while buffer2 is the one that I read. The checking of overrun and tx/rx ready seems to be done automatically and only accessible via interrupts rather than having specific CPU accessible status register bits, so that's what I mean in my original post, i.e. a polling based UART doesn't seem possible.

.