

Re: detecting characters on RS232–Interface

Source: <http://www.tech–archive.net/Archive/VB/microsoft.public.vb.general.discussion/2005–02/0024.html>

From: Jim Mack (*jmack_at_mdxi.nospam.com*)

Date: 01/26/05

Date: Tue, 25 Jan 2005 19:14:45 -0500

Peter wrote:

- > *Thanks Saga and Dick*
- >
- > *Dick, I extended your procedure to the following:*
- > *1.) read data into string variable*
- > *2.) use Instr() to check for UUUU, if its there check whether the*
- > *remaining amount of chars is >= the message length. If the remaining*
- > *chars do not hold the message then read more data and append to*
- > *string. Get message out of string and clear the string.*
- > *3.) if Instr() in 2.) did not find the UUUU then check using right\$()*
- > *whether the last char of the string is a single U(in case UUUU is*
- > *splitted at the end of the receive buffer). If it is a U then read*
- > *more data and goto 2.) If there is no U at the end clear the string*
- > *and goto 1.)*
- >
- > *Did I forget something?*
- >
- > *About OnComm Event:*
- > *If RThreshold is set to 1 then the event will fire on every received*
- > *character? Is this recommended?*
- > *As data will be transmitted continuously all the time, wouldn't it be*
- > *better to set RThreshold to, say, double length of the valid data?*
- > *Or what about not using OnComm event and just periodically read the*
- > *data using Inbuffercount(>0 and Input() Method?*
- >
- > *One more question arised:*
- > *The garbage data can be any binary data while the valid data is text.*
- > *Would I be able to detect the UUUUs inside the binary garbage data if*
- > *InputMode=0 (text)?*
- > *Regards*
- > *Peter*

Dick has given you good advice (no surprise). What I usually do in this situation is to set up a simple state machine. All that means is that you have a static variable that contains a value indicating where in the process you are, and a 'receiving buffer' to which you append the comm input buffer.

In a loop, peel characters off front of the receiving buffer 1 by 1, making it shorter. If you reach the end, set it to "".

State 0 (the initial state) is where you start. Check to see if your char is a U. If it is, enter State 1 (set the state variable = 1). If not, stay in state 0 (do nothing).

While in State 1, examine the next char in turn. If it is NOT a U, revert to state 0. If it is, enter state 2.

Proceed through state 4 in the same fashion. When you reach this state you know that you have seen 4 consecutive U's.

In state 4, if the next char is a U, either stay in state 4 if it's OK to see more than 4 U's, or revert to state 1 (seen 1 U). If it's not a U, clear the output buffer by setting it = this char, and enter state 5.

In state 5, check each incoming char. If it's an illegal char, dump the buffer, issue an error, ignore it, or whatever you like. If it's a terminating char, place the output buffer in a known location, raise an event or call a processing function, and enter state 0.

This is basically just Select Case statements driven by incoming characters -- the outer case is the state variable. When a character arrives, the machine answers the questions "what am I expecting?" and "what do I do with what I got?".

Using a state machine means you can completely define the behavior of the code and allow it to react to any possible character arriving at any time.

--

Jim Mack
MicroDexterity Inc
www.microdexterity.com