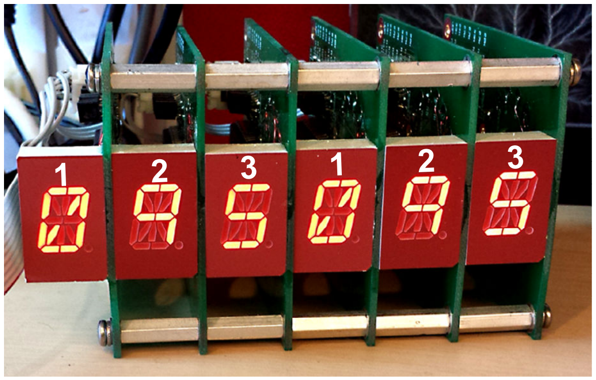
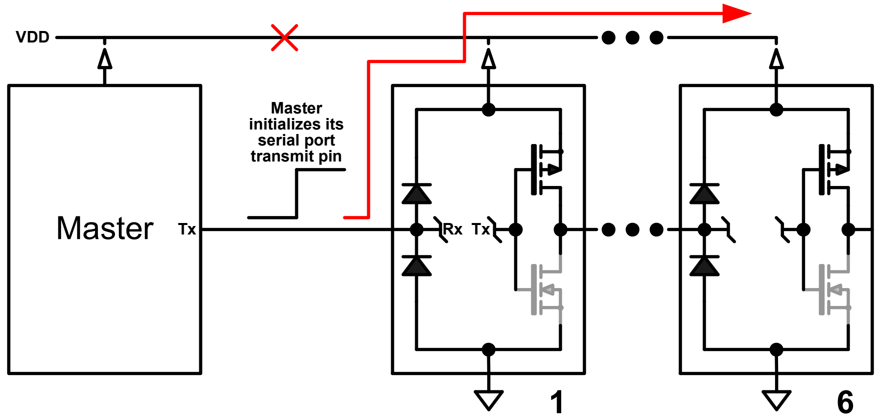
B7971 Smartsocket Enumeration

Correctly enumerated: “09:48:54”



Incorrectly enumerated: “09:50:14”



Module 1 can be forced to be the master by starting it just slightly faster than the rest of the units: It receives its power a few microseconds earlier from its internal structure than every other module in the chain. Not a reliable solution!

For this discussion, the left most module is connected to the master controller and has some number of modules connected to its right.

At power up all of the modules transmit 0xAB to the right and then after a delay, check their receive buffer. If the receive buffer of a module contains 0xAB then it assumes that it has a module to its left, i.e. modules 2 to 255 in the chain. At this point they don't know where they lie, only that they are not the first module at the far left. A module that does not have 0xAB in its receive buffer after the delay, because it has no neighbors to its left, assigns the address of 1 to itself and then transmits that 1 to the module at its right. Each module that receives a byte adds 1 to this byte, assigns this new address to itself and transmits this address to the right. This way the entire chain is sequentially numbered and can now interpret the display data that it receives.

When you power up a string of units, if the timing of the reset-test-enumerate sequence is off then the displays will assign an incorrect address to themselves and won’t be able to tell which part of the display data they receive belongs to them resulting in mixed display results. On my test bed with 6 digits, about 1 of 8 power cycles (Plugging into the USB to serial converter) results in the enumeration process failing: The upper picture is the correct time being displayed where the lower picture shows digit 1 correctly enumerating as "1" followed by "2" and "3" but digit "4" incorrectly enumerates as "1" forcing the modules to its right to addresses "2" and "3" for digits 5 and 6 respectively. This means that digit 4 found a byte that was not 0xAB in it's receive buffer telling it that its address was "1".

What is happening is that when the receive hardware is turned on, if the Rx pin is a logic low (Anything below 1/2 VDD) this will be seen as a start character and the rest of the byte will be built asynchronously. Even if a module to the left transmits a 0xAB, the Rx buffer already has whatever random data (Most likely not 0xAB) is in its buffer and assumes that it is the leftmost module since that data is non 0xAB.

What you are accomplishing by removing the 5V rail from the equation is to power the leftmost module ALONE via its receive pin ESD structure, slightly delaying the power up of each subsequent module and forcing the chain to enumerate properly, sometimes! (There is not brownout so these processors will operate down to 2.00V).

From the source code:

```
;Serial data is handled in the interrupt. A received byte sets the interrupt flag
;and immediately after receiving it it is retransmitted to the next element
;in the chain. The byte is then tested for validity. The receipt of character 13
;<ENTER> causes the program to branch, while in the interrupt, to establish
;what type of message has been received and whether or not it is intended for
;this element. For example, the receipt of an exclamation mark means that
;this element can ignore the data.
;
;The IC's communicate with one another in simplex format at power up to allow
;each element to determine it's own position within a chain of devices.
;The most important device is device number 1. Device number 1 enumerates
;itself with number one and then passes it's number to the next element
; in the chain. The next element increments the number it received and
;assigns itself to the new number...and so on up to 255 elements.
;The way the first element determines that it is in fact the first element
;is by it looking in the serial USART receive buffer at power up, and if it
;does NOT contain '171' it is the first element.
;
;At power up all devices clear their input buffer, pause briefly, and then
;transmit '171' to the next device in the chain. Therefore all elements
;will have 171 in their receive buffer with the exception of the first
;device.
```