READ PORT OPERATION OF THE 3.3V MULTI-QUEUE FLOW-CONTROL DEVICE

By Stewart Speed

APPLICABLE DEVICES

IDT72V51233, IDT72V51243, IDT72V51253
IDT72V51333, IDT72V51343, IDT72V51353
IDT72V51433, IDT72V51443, IDT72V51453
IDT72V51543, IDT72V51553
IDT72V51236, IDT72V51246, IDT72V51256
IDT72V51336, IDT72V51346, IDT72V51356
IDT72V51436, IDT72V51446, IDT72V51456
IDT72V51546, IDT72V51556

INTRODUCTION

The purpose of this application note is to provide the user with some idea of how the data being read from a multi-queue flow-control device may be handled by a "reading device", such as an FPGA or any device that controls the read port of the Multi-Queue. There are a number of reading scenarios the user may want to consider, here we try to look at each scenario in turn and attempt to provide some idea as to how the reading device may handle the data.

The device controlling the read port is required to be capable of recognizing that a valid data read, (that is to say that a new data word may have been accessed and placed onto the Multi-Queue output bus), may have occurred due to at least one of four possible events. Here is a description of the four possible events:

1. A read may occur from a previously selected queue by taking the $\overline{REN}$ input LOW. Note, here that a new word is accessed on the rising edge of read clock and is available for the reading device to process that word on the following rising edge of read clock.

2. The Multi-Queue device operates in a "First Word Fall Through", FWFT mode. One implication of this is that if the same queue is selected on both the write and read ports the first word written into the queue will automatically fall through to the outputs, regardless of the state of $\overline{REN}$.

3. During a queue switch on the read port a final read will occur from the old queue. Data from the old queue will be accessed and placed onto the Multi-Queue data outputs regardless of the state of $\overline{REN}$. This is due to the "Next Word Fall Through" nature of the device. This word from the old is forced out to allow an automatic read from the new queue, where the next word available in the new queue falls through to the outputs of the Multi-Queue. This leads to event four.

4. Also during a queue switch on the read port, the first word from the newly selected queue will be accessed and placed onto the Multi-Queue data outputs regardless of the state of $\overline{REN}$. Again this is due to the "Next Word Fall Through" operation, where the next word available in the new queue falls through to the outputs of the Multi-Queue.

This Application Note reviews these four events and shows an overall solution that may be implemented into the control logic of the reading device to handle any event.

READ OPERATIONS ON THE CURRENT QUEUE

Figure 1 illustrates the potential event outlined in item 1 above. The diagram shows the situation where a queue has previously been selected on the read port and read operations are occurring based on valid words being available in the queue and the $\overline{REN}$ input being toggled LOW to read out data words.

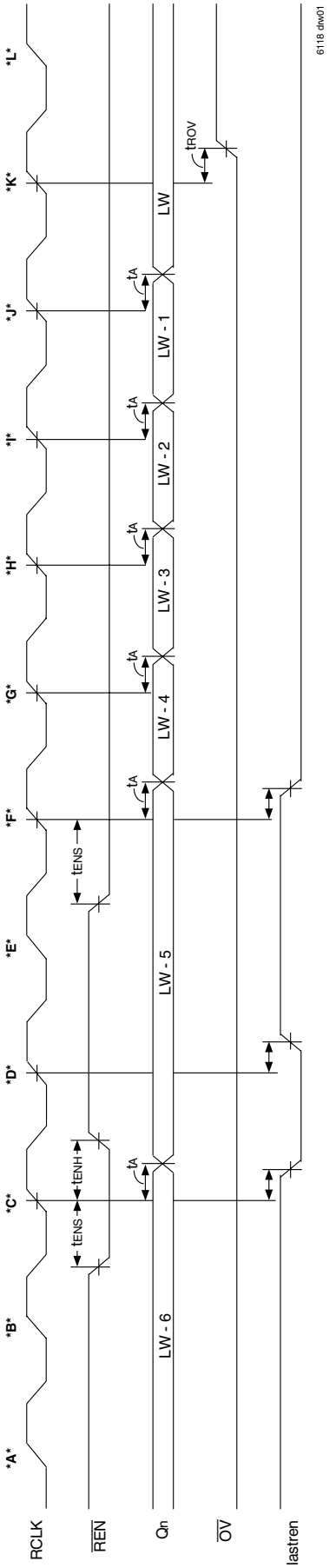
The Output Valid flag is essentially provided to indicate when a queue is empty or has been read to empty. It does not indicate when a new word has been placed on to the data outputs.

This diagram illustrates the implementation in the reading devices control logic of a flip-flop called "lastren", that indicates when an enabled read has been performed and the word on the output bus should be processed by the reading device. For example, if we look at cycle ***C*,** we can see an enabled read is performed and word "LW-5" is placed onto the outputs. On the next the cycle, cycle ***D*** the reading device should process this word "LW-5". The state of "lastren" is used to determine whether the word is a new word and therefore whether it should be processed.

Based on this set-up the reading device will process words on the following cycles:

D*, *G*, *H*, *I*, *J* & *K*.** Note, that on cycle ***L the output valid flag, $\overline{OV}$ is HIGH, therefore the queue has essentially been read to empty.

In summary, when a normal read event occurs on the current queue, a new word should be processed by the reading device on an RCLK cycle where the $\overline{OV}$ flag is true, LOW and the "lastren" flip-flop is true, LOW.



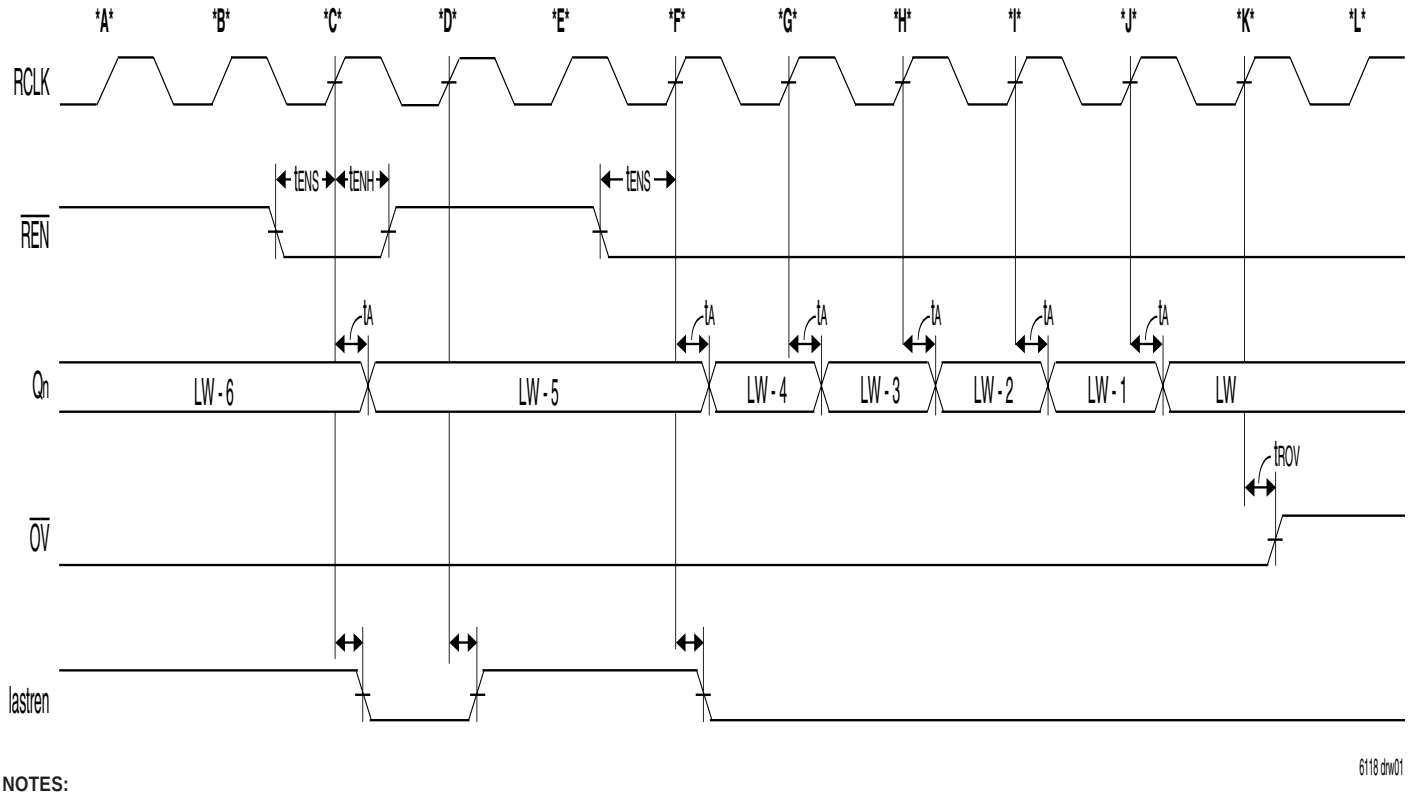
NOTE:
1. Normal operation. Queue already set-up, $\overline{REN}$ toggles to access data.
2. LW is Last Word.

Figure 1. Normal Read Operation

READ OPERATION DUE TO FWFT IN CURRENT QUEUE

Figure 2 illustrates the "First Word Fall Through" (FWFT) effect, this the second possible read event mentioned earlier. In this diagram both the Write and Read ports are selected for the same queue. A new word, "W1" is written into that queue, which before the write operation was an empty queue, the $\overline{OV}$ flag is HIGH. This first word written into the queue automatically falls through to the data outputs, causing the $\overline{OV}$ flag to go active, LOW. Note, that this first word has fallen through regardless of the state of $\overline{REN}$, in fact $\overline{REN}$ is HIGH throughout this diagram. Therefore, the use of the "lastren" flip-flop mentioned above is no longer an option to determine if a new word is available for the reading device to process. We have now included a second flip-flop, "lastov" into the control logic of the reading device. This flip-flop provides the reading device with a status of the $\overline{OV}$ flag delayed by one read clock cycle. Therefore, the reading device monitors both $\overline{OV}$ and "lastov" to determine whether a new word is available, here we can see that on read clock cycle *C* the first word, W1 should be processed by the reading device.

In summary, in the case of the FWFT read event a new word should be processed by the reading device on an RCLK cycle where the $\overline{OV}$ flag is true, LOW and the "lastov" flip-flop is false, HIGH.



NOTES:
1. The same queue is selected on both the write port and read port.
2. $\overline{OE}$ is LOW.
3. The First Word Latency = t_{SKEW1} + RCLK + t_A. If t_{SKEW1} is violated an additional RCLK cycle must be added.

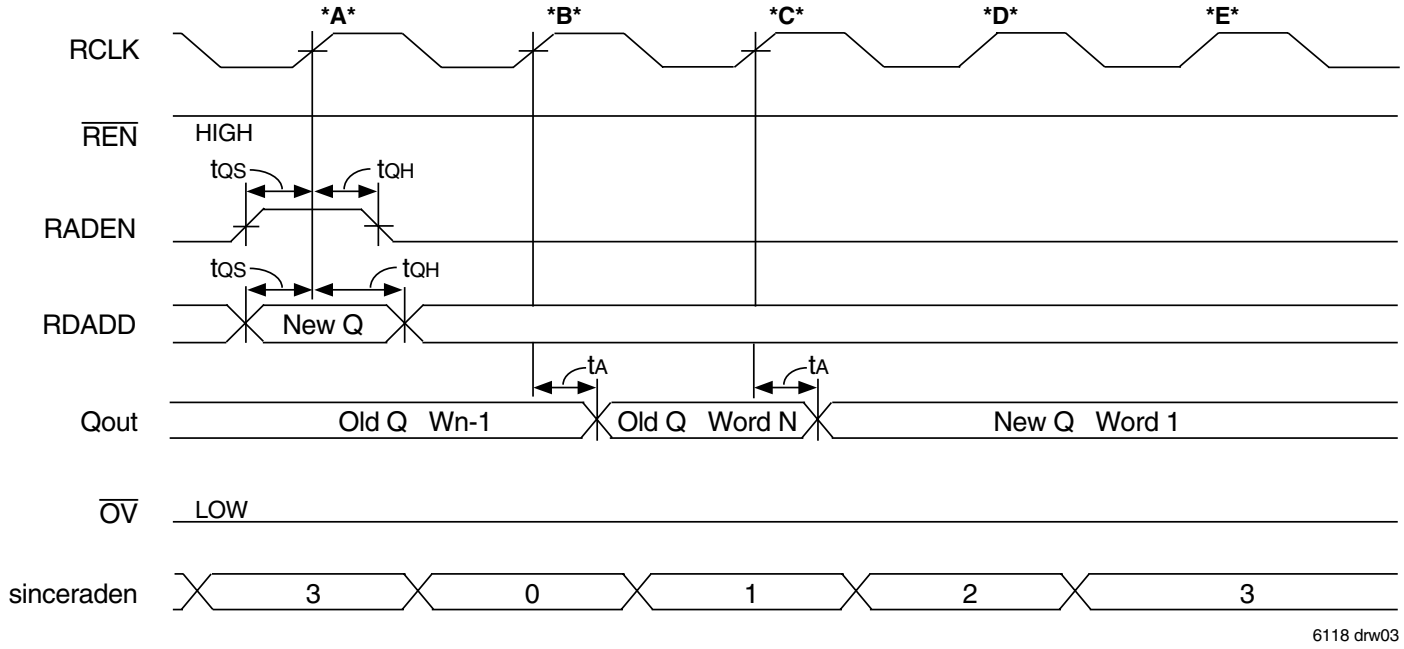
Figure 2. Read due to FWFT

READ OPERATION DUE TO A QUEUE SWITCH ON THE READ PORT

Figure 3 illustrates the third and fourth possible read events, which is indirectly due to the "First Word Fall Through" effect caused by a queue switch on the read port. In the diagram we can see that a "New Q" is selected on the read port on RCLK cycle ***A***. Here RADEN is HIGH and the RDADD address bus is addressing the new queue. When a queue selection is made the first word of the new queue will fall through to the data outputs of the read port, forcing a final word to be read from the previous (old) queue. This occurs regardless of the state of $\overline{REN}$, in fact the $\overline{REN}$ input is HIGH throughout this diagram.

During a queue switch the $\overline{REN}$ input can remain HIGH and the $\overline{OV}$ flag can remain LOW indicating that both the old queue and the new queue have not been read to empty. Therefore, we introduce a register "sinceraden" into the control logic of the reading device. Monitoring of the "lastren" and "lastov" flip-flops alone does not indicate to the reading device that a new word is available in this example, hence the introduction of the register, "sinceraden". This register is a 2 bit register and in the event of a queue switch is simply used to provide a count from 0 through 3. Upon RADEN going HIGH this register should be reset to a value 0. On the following three RCLK cycles this count will be incremented up to a value of 3, at 3 the count will cease incrementing and wait until RADEN is toggled HIGH once more.

In the event of a queue switch a minimum of 2 new words will be automatically read out from the Multi-Queue device and will need to be processed by the reading device, (this again is an effect of the "Next Word Fall Through" operation). From the diagram we can see that on RCLK cycle *B* "Word N" is read from the "Old Queue". This word must be processed by the reading device on RCLK cycle *C*. The same is true for the first word of the New Queue, "Word 1" which must be processed by the reading device on RCLK cycle *D*. The reading device should monitor the "sinceraden" register and when the register has a value of either 1 or 2 the data word on the bus is valid, providing of course that the $\overline{OV}$ flag is LOW. It is also very important during a queue switch on the read port, that the reading device realize that the new word processed with "sinceraden" at value 1, is data from the old queue, and therefore processed appropriately. Secondly, the new word processed with "sinceraden" at value 2, is data from the new queue, and therefore processed appropriately.



NOTES:
1. Sinceraden is a 2 bit register.
2. Sinceraden should reset to 0 when RADEN is HIGH.
3. Sinceraden should increment to 3 and stop incrememting, then wait for RADEN to go HIGH again.

Figure 3. Read due to Queue Selection

AN EXAMPLE OF CONSECUTIVE READ EVENTS

Figure 4 shows an example where are number of read events occur consecutively. From Figure 4 we can see that an enabled read has occurred on the same RCLK cycle as the queue switch, cycle ***A***. In this example the "sinceraden" register serves exactly the same purpose as before, indicating in conjunction with the $\overline{OV}$ flag that a new word must be processed on RCLK cycles ***C*** and ***D***, ("sinceraden" = 1 or 2). However, in this event we must also process a word from the old queue that was accessed on RCLK ***A***. This word from the old queue, word "N-1" must be process by the reading device on RCLK *B*. The reading device control logic processes this word due to the fact that the "lastren" flip-flop is active, LOW on this cycle, and in conjunction with $\overline{OV}$ being true processes the word "N-1". As with the previous example, it is important to note that the data words processed by the reading device on RCLK cycles *B* and *C* are from the old queue and are therefore processed appropriately.

In summary, when a queue switch is made it is possible that up to three words will need to be processed by the reading device. The first word will be processed by virtue of the fact that an enabled read was performed at the queue switch and therefore, the state of "lastren" in conjunction with the $\overline{OV}$ flag will ensure this word is processed. The following 2 words will be processed by virtue of the fact that register "sinceraden" has the value of 1 or 2, and that the $\overline{OV}$ flag is active, LOW. Again, it is important to note that a valid word read with "sinceraden" equal to '1', is a word from the old queue and therefore processed appropriately.

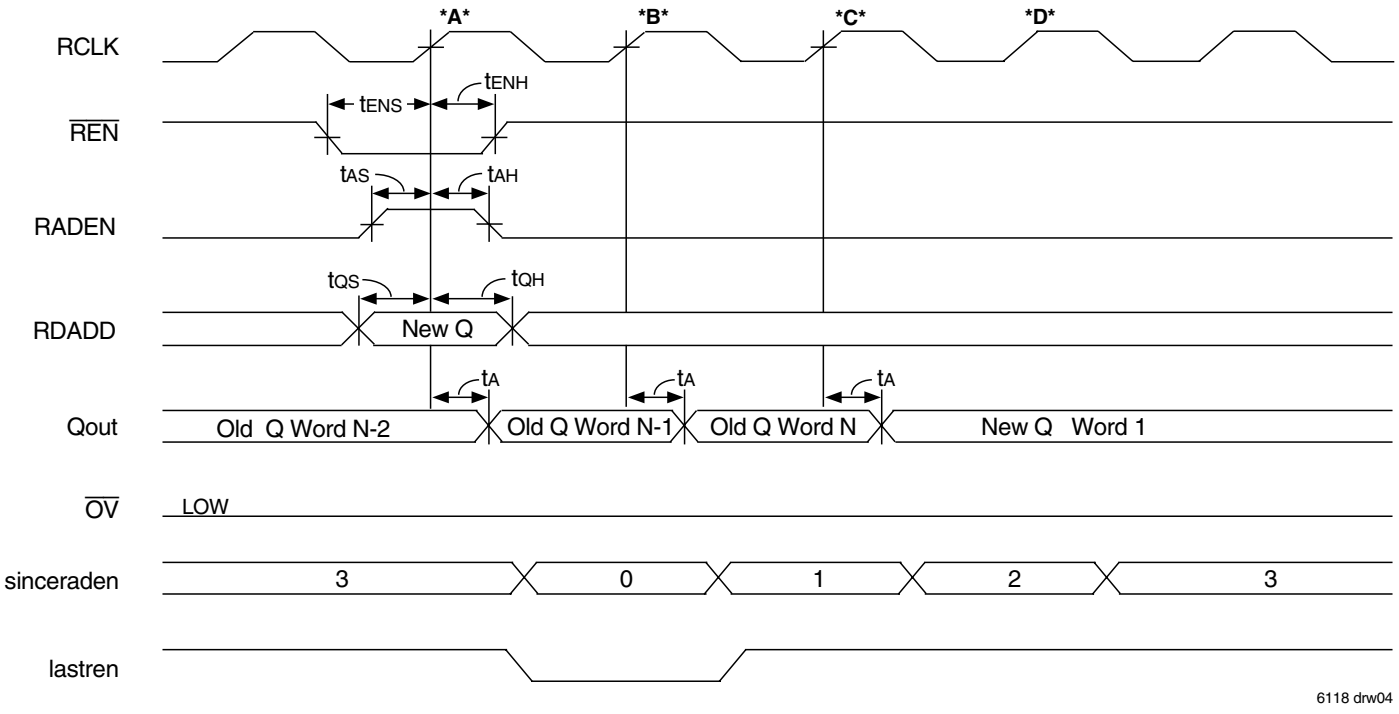


Figure 4. Multiple Reads Example

HARDWARE IMPLEMENTATION

The control logic required to handled all of the events outlined above can be easily implemented in the reading device (FPGA), below is sample of Verilog code that produces the control logic required to detect when a word out of the Multi-Queue read port is a new, valid word that must be processed.

FPGA - Verilog Code Sample

```
always @ (posedge rclk ) begin

        if( (!ov_) && (
                ( !lastren_             //Check for normal read operation
                || lastov_  //Check for FWFT independent of queue switch
                || (sinceraden == 1) //Check for old queue old during queue switch
                || (sinceraden == 2) )))        //Check for first word of new queue during queue change
                        begin
                                /*Here the user should implement code that handles a valid data word according to their application requirements.
                                Note, that sinceraden is utilized during a read port queue switch. The code outlined above only interprets when a
                                new, valid word has been read out of the Multi-Queue device.
                                If sinceraden <= 1 then the word is from the old queue.
                                If sinceraden >= 2 then the word is from the new queue. */
        end

        if( sinceraden < 3 ) sinceraden = sinceraden + 1;

        if( raden ) sinceraden = 0;     //sinceraden is a 2 bit register that is reset to 0 when
                                        //RADEN is HIGH, on a queue switch cycle

        lastren_ = ren_;                //REN flip-flop
        lastov_ = ov_;                  // OV flip -flop
end
```