1 Calculating delays for messages buffered in the regulator

The regulator uses a circular FIFO with two methods for accessing data: `fifoRead` and `fifoWrite`, which respectively read and write one element at a time, and a method for checking the current utilisation ie. how many elements are currently being stored, `fifoGetRelPos`.

As soon as a message is received, two conditions are checked: the time since the last message delivery $t_{in} - t_{\text{out}}^{\text{last}} \geq \Delta$, where Δ is the minimum inter-arrival time after regulation, and if the buffer is empty. If these two conditions are fulfilled, the message is immediately delivered, as for t_{out}^{n} in fig. 1.

If the inter-arrival time is $< \Delta$, but the buffer is empty, as for t_{in}^{n+1} in fig. 1, then the message must be buffered until it can be delivered. It is pushed to the FIFO and a subsequent pop is scheduled for the actual delivery time using an offset given by the remaining time until Δ has elapsed from the last delivery. This offset is given by $t_{out}^{last} + \Delta - t_{in}^{n}$.

If the inter-arrival time is $< \Delta$, but the buffer is *not* empty, as for t_{in}^{n+2} in fig. 1, then the message must be buffered until it can be delivered at a time after the previously scheduled deliveries. Again, it is pushed to the FIFO and the delivery is scheduled using an offset, this time given by $(t_{out}^{last} + \Delta - t_{in}^{n} + \Delta \cdot \text{BUF\_POS})$, where BUF_POS is the relative position in the FIFO. This means that the offset is shifted in time by multiples of Δ depending on how many messages are pending delivery.

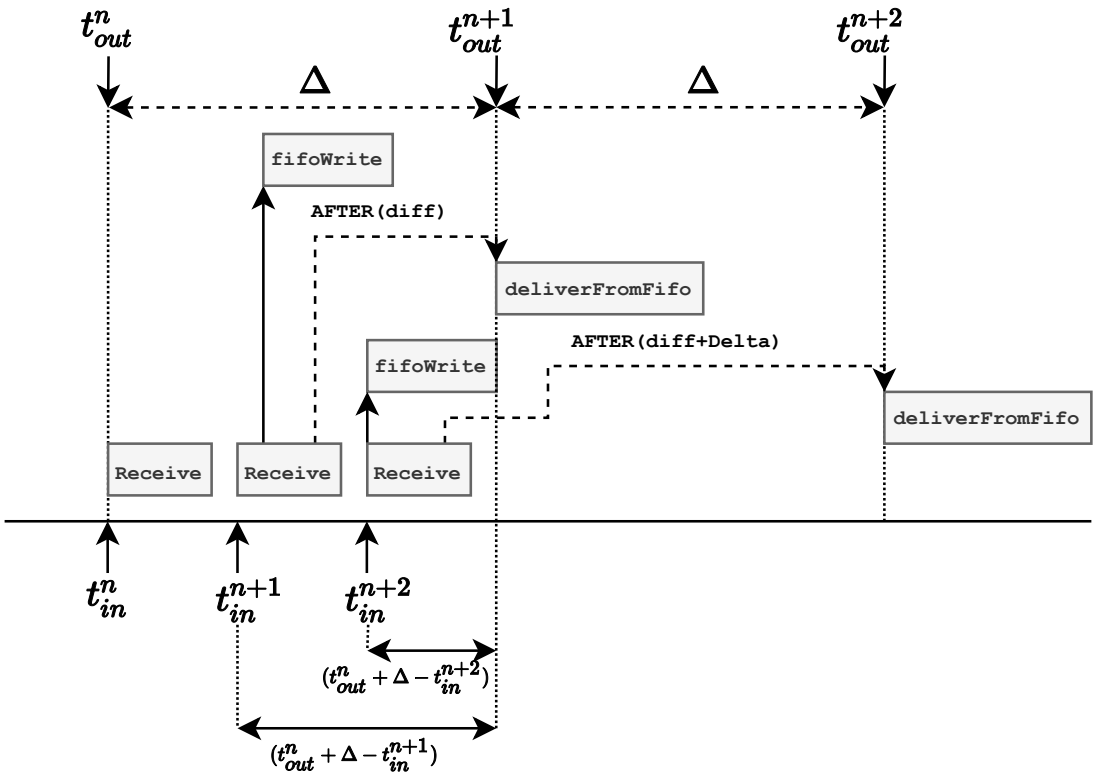

Figure 1: Timing diagram for CAN message regulator