The Schmitt trigger converts analog signals to digital signals by setting two different threshold voltages, an upper and lower threshold. When the analog signal is inputted, the trigger will output a high voltage, once it reads the upper threshold on input signal, and output a low voltage when it reads the lower threshold. This solution minimizes the amount of noise in the digital signal. The STM32 platform has its GPIOs equipped with Schmitt trigger inputs, with built-in upper and lower thresholds. When the GPIO reads an input above 70% of the power rail (3.3V) then that pin reads “high,” which is the upper threshold. As for the lower threshold, the GPIO will read a “low” when a voltage lower than 30% of the power rail is inputed.

We are using the schmitt trigger in our design to read frequencies of incoming signals from the radio. Then, we use the STM32’s internal timers to count up to a set period, which is how long the input signal will be sampled and how long it would take to transfer a bit. The set period will be dependent on the baud rate (1200 bps): set period = 1/(baud rate). We then create a variable to keep count of how many times the signal toggles from the upper to lower threshold or vice versa. Once the timer counts up to the set period, the frequency is calculated: Frequency = (toggle count \* Clock frequency)/(Clock Prescaler \* set period). If the calculated frequency is 2200 Hz it will generate a one and if 1200 Hz a zero will be generated. The ones and zeros can then be stored in an array, with the first element being the MSB. Then the microcontroller will use the CRC calculation unit to generate a CRC value to compare with the value in the FCS section in the packet. If the CRC value matches the FCS value, then the packet is valid and will go on with the transmission process. If it is not a valid packet, then it will not transmit the packet.

**Pseudo code:**

Initalize toggle\_count = 0, frequency, set\_period;

Timer.Instance->CNT = 0; //set timer to 0

while((Timer.Instance->CNT) < set\_period)

{

while ( GPIO reads low); *//wait for GPIO to read high*

while (GPIO reads high); *//wait for GPIO to read low*

toggle\_count++;

}

Frequency = (toggle\_count / (set\_period));



