### PDCCH Transmitter

The figure below depicts the PDCCH transmitter chain and its mapping to BCP resources.



Figure : PDCCH transmitter signal chain and processing resource mapping

1. The CRC sub module can perform block concatenation after attaching CRC to each block, but only one METHOD2\_ID can be specified which is used to mask all the attached CRCs. Since each PDCCH requests a different METHOD2\_ID, the CRC sub module cannot be used to perform CRC attachment for multiple PDCCHs with a single input packet and so the CRC attachment should be performed by the CPU.
2. ENC is a block-based sub module which supports at most three different block sizes (*K*’s) in one packet. The PDCCHs transmitted in one TTI may have more than three kinds of payload block sizes. For example, in the case of an FDD 20MHz cell with 4 antenna ports, as per section 5.3.3.1 of [3] we can figure out that there are at most six types of DCI payload size - 15 (format 1C), 28 (format 0/1A/3/3A), 33 (format 1B/1D), 39 (format 1), 50 (format 2A), 54 (format 2). So the BCP ENC needs 2 input packets to perform encoding for 6 PDCCHs in one TTI.
3. RM is also a block-based sub module. Unlike ENC, RM needs both the payload block size (block size before rate de-matching, denoted by K) and the channel block size (block size after rate de-matching, denoted by *E*) as it’s input parameters. There are four PDCCH formats and each format corresponds to one *E* value. In the above example scenario (FDD 20MHz cell and 4 antenna ports), there are at most 6x4=24 different [*K*, *E*] pairs so we need 12 BCP input packets to perform rate matching in one TTI with each packet processing the PDCCHs with two [*K*, *E*] pairs.

To fully use CRC, ENC and RM and keep simplicity of the framework, we recommend using one packet to process the CRC attachment, convolutional coding and rate matching of a single PDCCH.