

**Western Digital®**

# FTL Flow Control For CFexpress™ Camera Hosts Using Large NVMe™ Reads

Vishwas Saxena

Technologist, Firmware Engineering

August 7, 2019



# Agenda

1 CFexpress uses NVMe Commands with Higher MDTS

2 PRP Setup of Large commands

3 Need of FTL Flow Control for Large NVMe Reads

4 FTL Flow Control Design for Large NVMe Reads

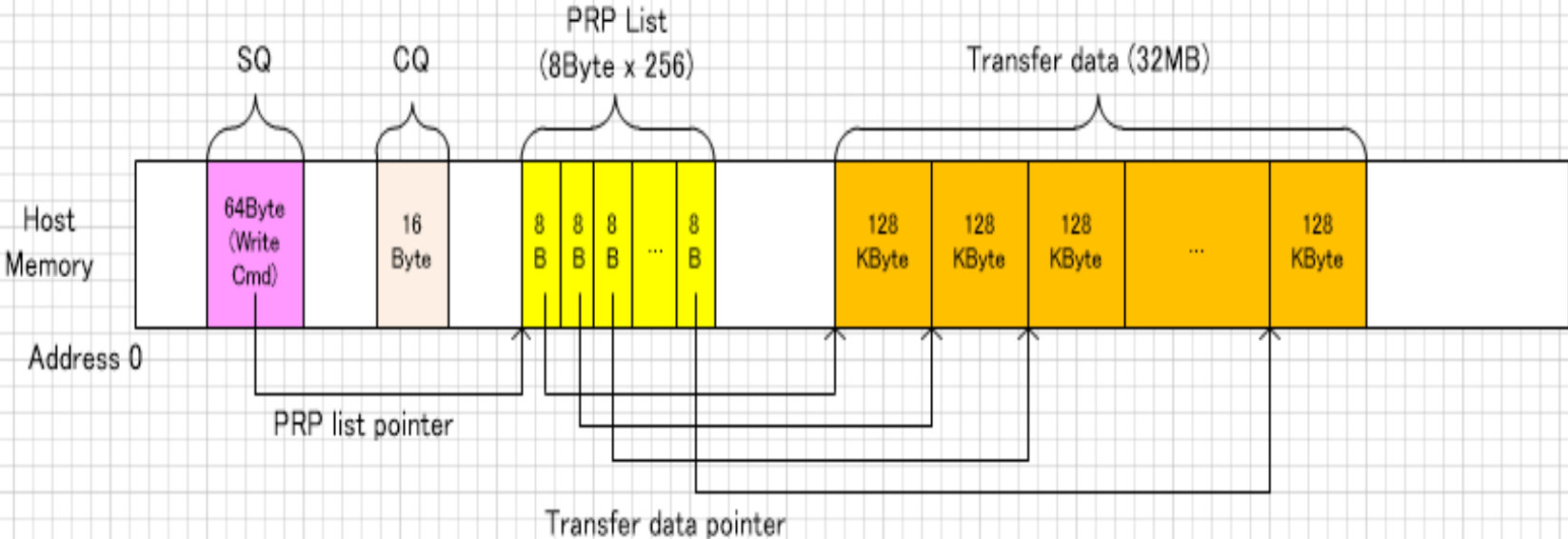
5 Summary

# Large NVMe commands

*CFexpress Cards support NVMe MDTS of 32MB*

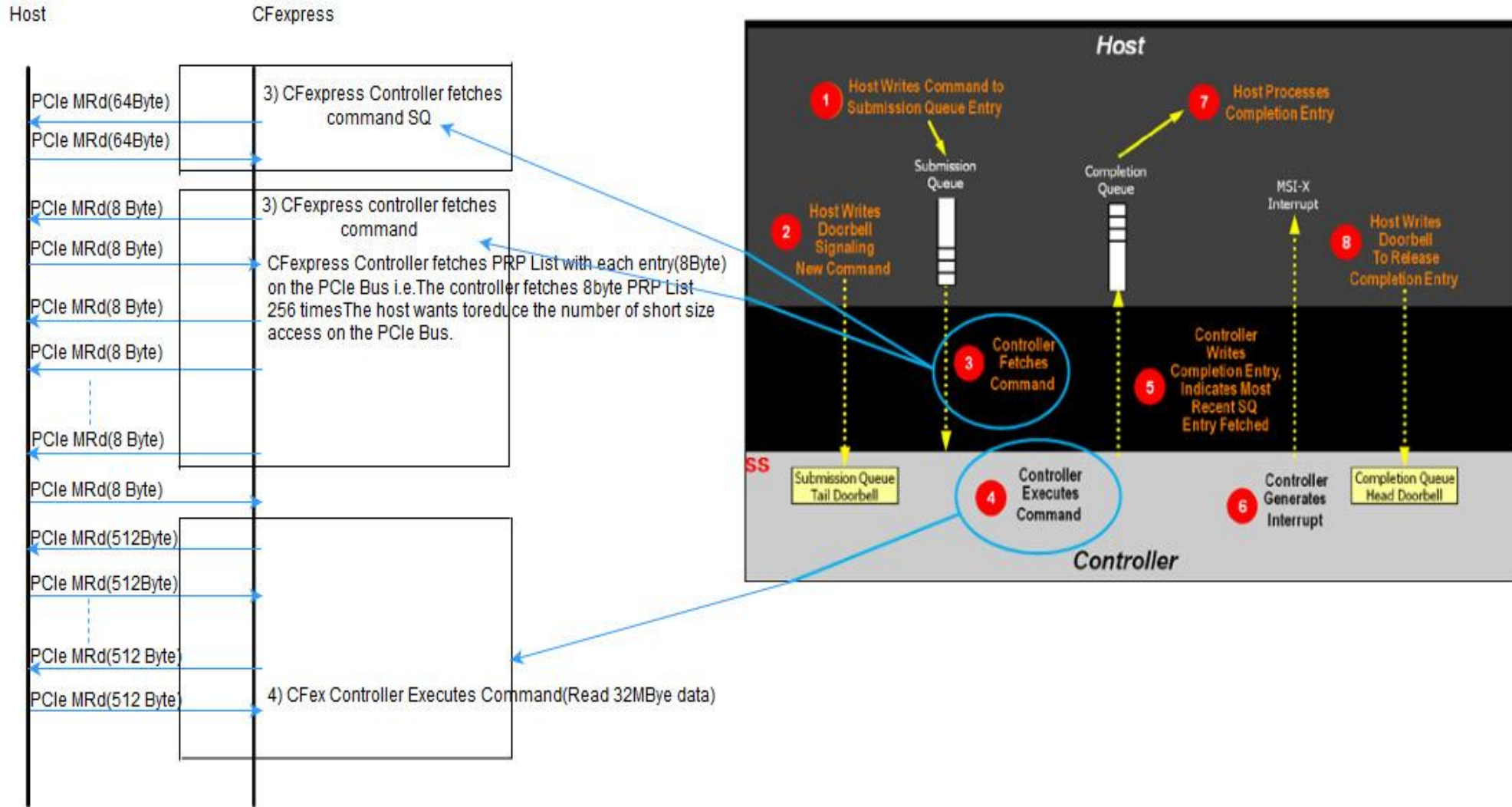
- Maximum Data Transfer size (MDTS)
  - MDTS indicates the NVMe maximum data transfer size between the host and the controller.
    - Host manages buffers of up to MDTS size
- Memory Page Size Maximum (MPSMAX) of 128KB
  - Host splits the 32MB NVMe command into 256 PRP entries of size 128KB
  - CFexpress card does prefetch of all 256 PRP entries
- Performance Impact of Large NVMe Command Size
  - Large NVMe command size results in higher performance due to shorter HTAT (Host Turn Around Time) on full card range

# 32MB Data Transfer in One Command



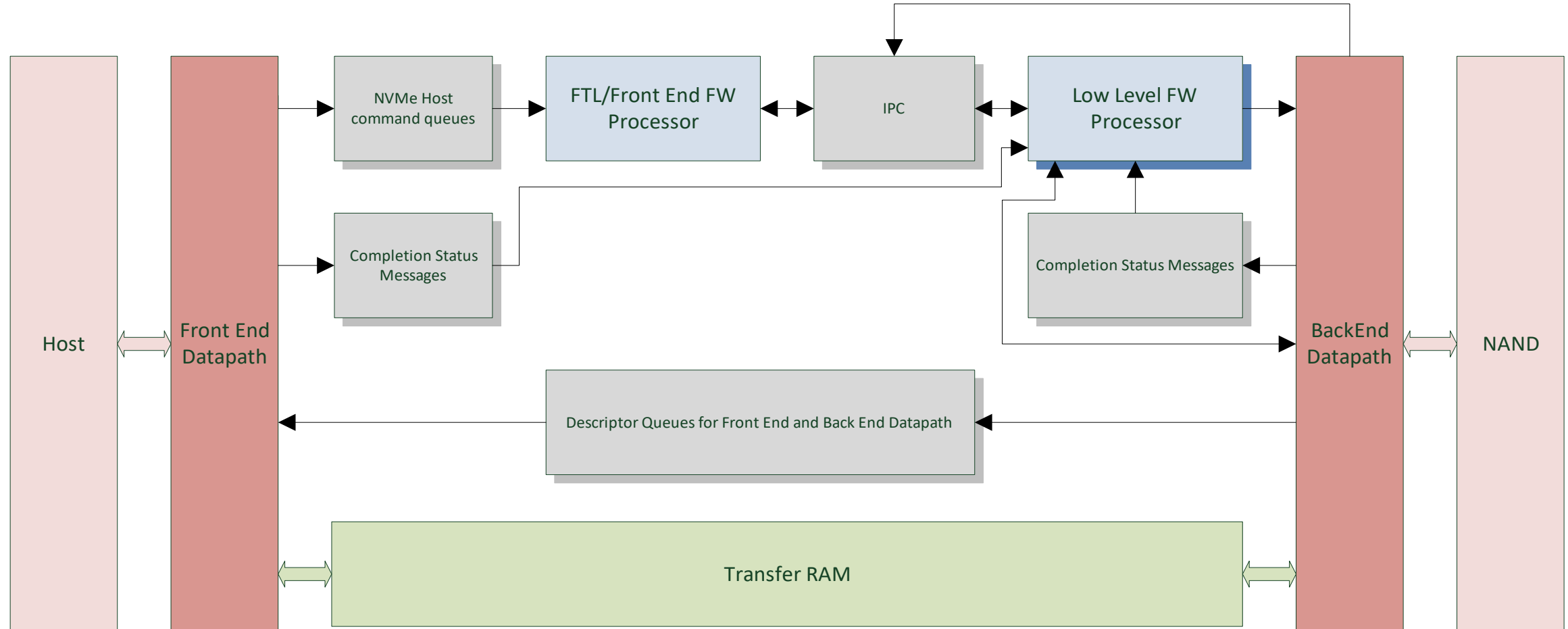
SQ: Submission Queue  
CQ: Completion Queue  
Memory Page Size: 128KByte  
Max Data Transfer Size: 32MByte  
Max Payload Size: 512Byte  
Max Read Request Size: 512Byte

# NVMe/PCI Bus Sequence Flow



# NVMe Flash Controller Architecture

*FTL and Low Level FW uses IPC Queues for non blocking FW architecture*



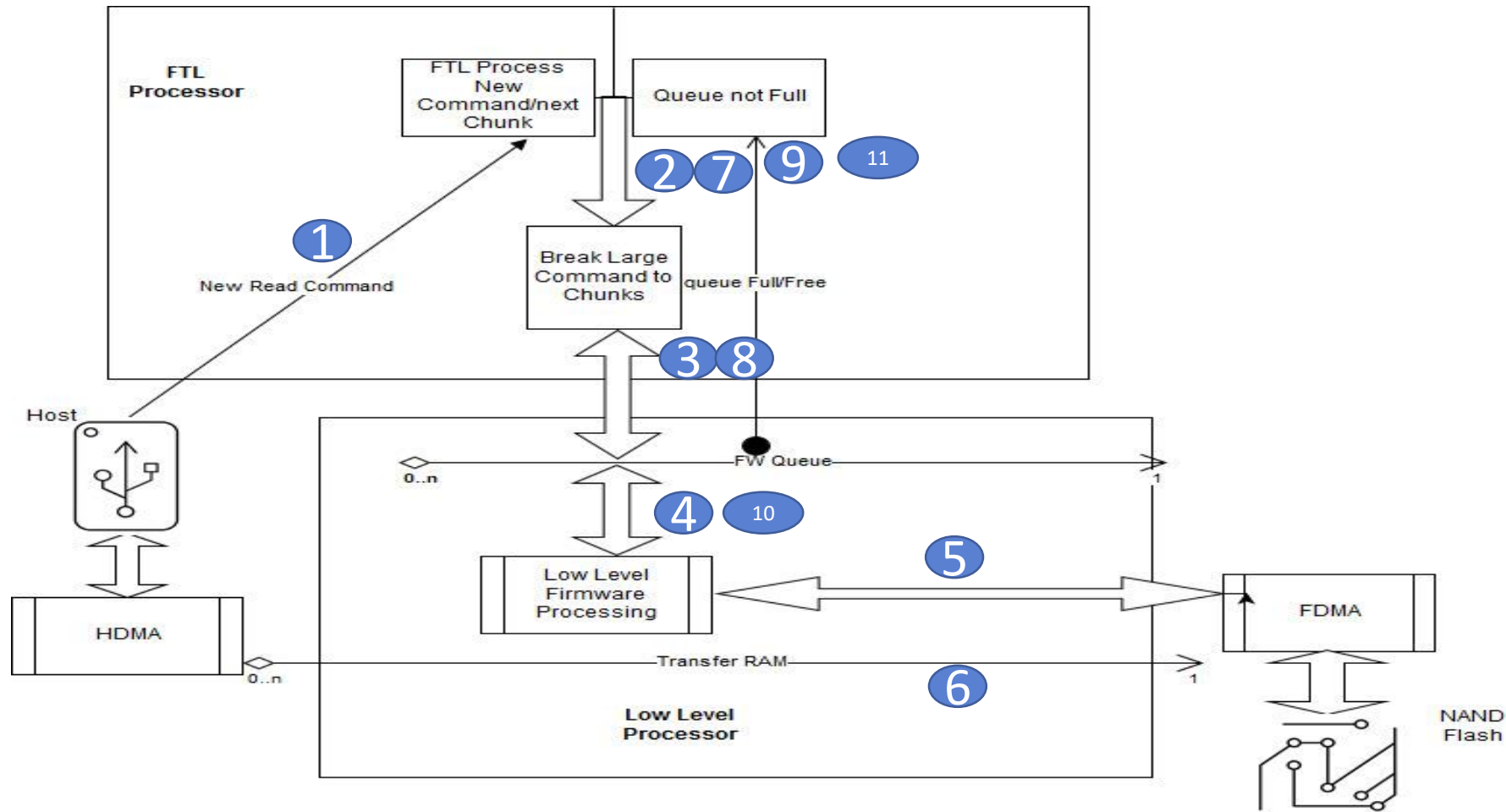
# Need of FTL Flow Control for Large NVMe Reads

*CFexpress Cards supports MDTs of 32MB*

- Series of FTL read requests from one 32MB command flood the FTL – Low Level FW IPC Queue
  - Typical NVMe device Flash controllers break down the NVMe sequential read command to die size(32KB) at FTL
    - FTL queues them to Low Level Firmware Queue (flash sequencer) with typical queue size of double the number of dies
    - Typically Queue size is 32 in 16 die NAND product
    - One 32MB size read command is broken down to 1024 requests that cannot be held in the above queue
  - This results in queue full scenario that causes stall in FW processing
  - Same queue is used by Low-Level Firmware (Physical Sequencer) and should always be free
    - To report error status to FTL
    - To get parity information location of failed physical page from FTL
- FW Queue Size Increase
  - Device Firmware cannot define FW Queue size (between FTL and Low Level Firmware) for such large commands
  - Larger Queue size increases the SRAM budget
  - With increase in Queue size we cannot avoid the problem as continuous burst of large reads of size 32MB will eventually exhaust the FW Queue entries

# FTL Flow Control

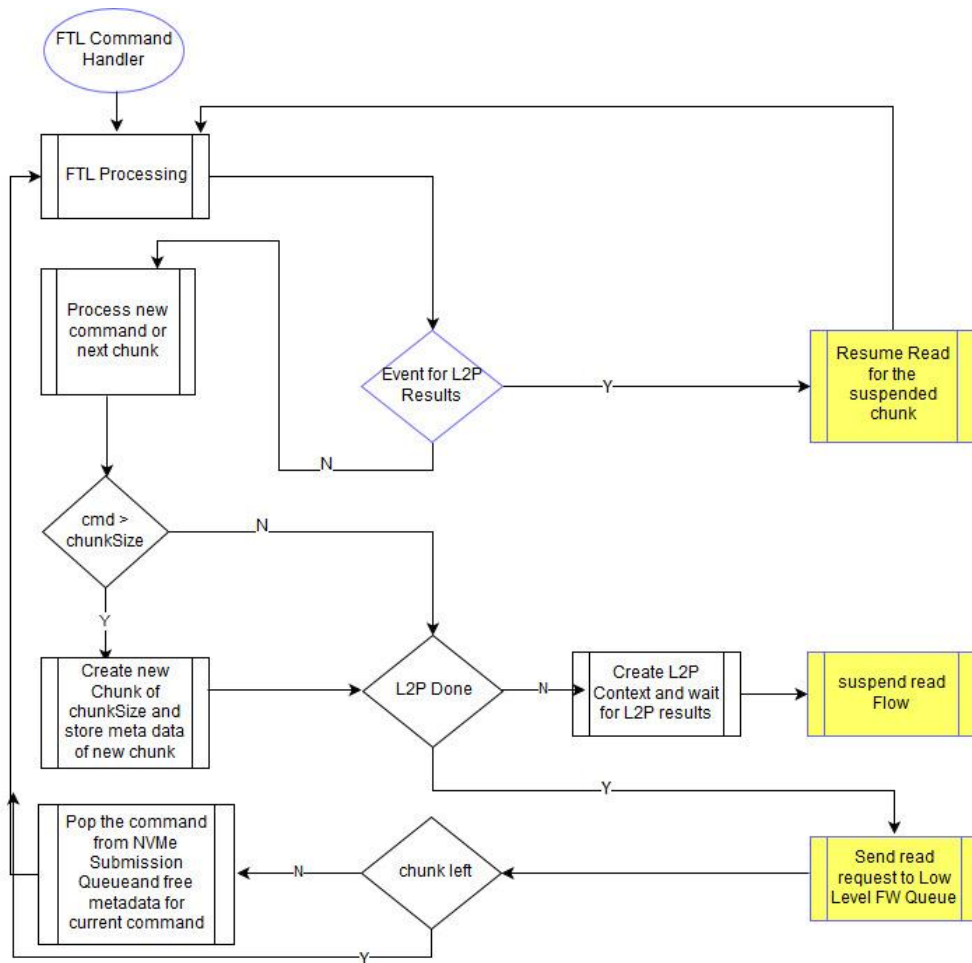
High Level Firmware Architecture with FTL Processor and Low Level Firmware Processor





# FTL Large Reads Flow control

## Flow chart



# Flow Control Solution


## *Detailed Steps*

- Break large commands into smaller chunks from FTL
  - Define chunk-size (configurable) based upon optimal Device Queue FULL threshold
  - Break the large sequential command into chunks of chunk-size
    - Do not pop the next command from NVMe submission queue till all chunks of current command are processed
    - All subsequent chunks will now come via FTL Processing for one 32MB command
  - Blocks FTL processing when FW Queue usage reaches QUEUE\_FULL threshold.
    - FTL processing of new command and next chunks is suspended unless queue full condition is cleared by the Queue ISR (low level FW) has consumed the entries
- Mixed Read workloads with short random commands
  - Allow processing the short random commands that are less than chunk size while processing current large sequential command in FTL processing
- Error Handling
  - Since same FW Queue is used to process error handling requests and read/write request and the design ensures that the Low Level Queue is not blocked so FW can complete error handling

# Summary

## *FTL Flow Control*

- Large Read commands are broken by FTL into chunks of configurable size before getting enqueued at FW Queue
  - FW Queue driver gives interrupt on threshold condition before queue full and queue available
- FTL Blocks processing of NVMe submission queue and internal chunks to ensure Flow control
- FTL optimizes the short random reads processing along with large sequence reads
  - Results into optimal performance in mixed Read (Sequential + Random) workloads
- FTL Flow Control allows error handling between FTL and low level Firmware to remain unaffected
  - As Low Level Firmware Queue is always free

- 
- Western Digital and the Western Digital logo are registered trademarks or trademarks of Western Digital Corporation or its affiliates in the US and/or other countries. The NVMe word mark is a trademark of NVM Express, Inc. All other marks are the property of their respective owners.



# **Western Digital<sup>®</sup>**

**Architecting Data Infrastructure for the Zettabyte Age**