Explanation:

We implemented two circle buffers for each device: encoder and decoder. We solved the synchronization problem with help of semaphores and its wait queue. For each device we have two counters for number of readers and number of writers. And also two binary semaphores: the writers semaphore initialized as unlocked and the readers semaphore initialized as locked (because in this case the reader should wait for the first writer that should unlock the semaphore for him). In each write() the writers semaphore we lock the writers semaphore at the beginning, then we do all writing operations and then the semaphore is raised. At the end we do the edge case checks. If the free size in buffer is 0 now then we lower the semaphore in order to put into the wait queue the next writer that do not have any space to write. When this writer will arrive, he will check is there any readers are active and then he will enter the wait queue. Otherwise he will return with EOF. Also writer should raise the semaphore for readers in case that there was no any data on start in the buffer and there were some data written in current call of the write(). In read function the use of semaphores is almost the same. The difference is that reader lower its semaphore in case of no data in the buffer and it raises the writers semaphore in case of no free space on start and some data was read from the buffer.