Note: This mailbox pattern does not exactly match the Queue-Mailbox Pattern FreeRTOS guide. I modified it for my own needs. Master/Slave Relationship QueueHandle txQue MSTCmdRequestQue = xQueueCreate(1, sizeof(SLV Response *)); SLV_Request *ptrSLVRequest = new Slave_CmdRequest(); SLV Response *ptrSLVResponse = new Slave Response(); xQueueHandle xQue SLVCmdRequests = xQueueCreate(1, sizeof(Slave CmdRequest *)); SLV CmdRequest *ptrSLVCmdRequest = new Slave CmdRequest; SLV Response *ptrSLVResponse = new Slave Response; Pointer to Slave's Queue / sending Slave Command SLV Cmd Request Queue ptrSlaveCmdRequest->QueueToSendResponse = xQue MSTCmdRequestQue ptrSlaveCmdRequest->Cmd = SLV COMMAND::Cmd1; xQueueSend(xQue SLVCmdRequests, &ptrSLVCmdRequest, 0); Pointer to Master's Queue / sending Slave_Response xQueueReceive(xQue_SLVCmdRequests, &ptrSLVCmdRequest, pdMS_TO_TICKS(300) < Act on ptrSLVCmdReguest->Cmd > Cmd Request Queue ptrSLVResponse->ResponseCode = SLV STATUS::Ok; may be someone else's Slave ptrSLVResponse->JsonResponse = new std::string(Data); // Populate response data xQueueSend(ptrSLVCmdRequest->QueueToSendResponse, &ptrSLVResponse, portMAX_Delay); SLV Response Queue enum class SLV_COMMAND : uint8_t None, Cmd1, Using a Queue as a mailbox significantly reduces the cost of Cmd2 memory, but at the expense of some time as two objects must coordinate when a command is sent and the response is generated. Typically the master will need to wait for a enum class SLV STATUS: uint8 t response. When the master doesn't need a response AND the data structure is small, a normal gueue can be created at OK. **ERROR** the Slave. Note: All the data structures are defined in terms of the Slave struct Slave CmdRequest for consistency. The pattern requires that Master have a separate Response Queues for each and every slave who QueueHandle t QueueToSendResponse, responds to a command because the returning data is formed SLV COMMAND Cmd to match the Slave's needs. struct Slave Response Notice that there are matching Request/Response structures on both sides. RTOS copies data between the two for you. SLV STATUS ResponseCode, Care must be taken when working with dynamic data for std::string *JsonResponse which RTOS only copies your pointers. Slave Responding Object Master Calling Object