FHTX和FHRX流程设计和分析 V1.0

目录

[FHTX和FHRX流程设计和分析 V1.0 1](#_Toc29236110)

[修订记录 3](#_Toc29236111)

[1 FH TX 处理设计 4](#_Toc29236112)

[1.1 FH TX 内存分配 4](#_Toc29236113)

[1.2 FH TX buffer填写 5](#_Toc29236114)

[1.3 FH TX memset的处理 6](#_Toc29236115)

[1.4 需要注意的问题 7](#_Toc29236116)

[1.4.1 发送到FPGA的数据是否进行32byte对齐？ 7](#_Toc29236117)

[1.4.2 sFrontHaulTxBbuIoBufCtrl[bufIdx][nCellIdx].sBufferList.pBuffers[i]中的i使用symbol是否合理？ 7](#_Toc29236118)

[2 FH RX处理设计 7](#_Toc29236119)

[2.1 FH RX接收的回调函数cpa\_fh\_rx\_callback 9](#_Toc29236120)

[2.2 FH RX 内存分配 9](#_Toc29236121)

[2.3 FH RX 接收到的数据buffer填写到X86 10](#_Toc29236122)

[2.3.1 Pusch的处理 10](#_Toc29236123)

[2.3.2 DMRS的处理 11](#_Toc29236124)

[2.3.3 PUCCH(待补充) 11](#_Toc29236125)

[2.3.4 PRACH(待补充) 11](#_Toc29236126)

[2.3.5 SRS(待补充) 11](#_Toc29236127)

修订记录

|  |  |  |
| --- | --- | --- |
| 修改内容 | 作者 | 修订日期 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# FH TX 处理设计

FH TX发送的数据格式如下：

|  |  |
| --- | --- |
| FH header 32Byte | |
| Symbol0 | ANT 0 IQ |
| ANT 1 IQ |
| FH header 32Byte | |
| Symbol1 | ANT 0 IQ |
| ANT 1 IQ |
| … … | |
| FH header 32Byte | |
| Symbol13 | ANT 0 IQ |
| ANT 1 IQ |

## FH TX 内存分配

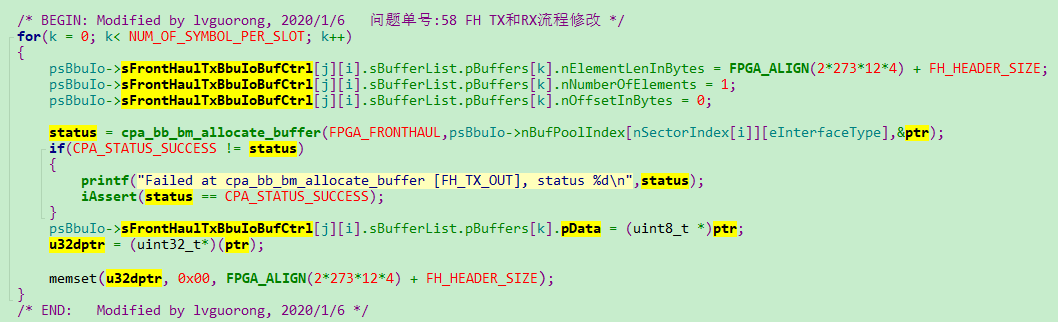
FH TX的内存分配在init\_fpga函数中进行处理：

1、先进行初始化，用来判断是否有足够内存可以分配：

|  |
| --- |
| cpa\_bb\_bm\_init (FPGA\_FRONTHAUL, &psBbuIo->nBufPoolIndex[nSectorIndex[i]][eInterfaceType],  N\_FE\_BUF\_LEN\*NUM\_OF\_SYMBOL\_PER\_SLOT, FPGA\_ALIGN(2\*273\*12\*4) + FH\_HEADER\_SIZE); |

2、通过函数cpa\_bb\_bm\_allocate\_buffer实现：

|  |
| --- |
| cpa\_bb\_bm\_allocate\_buffer(FPGA\_FRONTHAUL,psBbuIo->nBufPoolIndex[nSectorIndex[i]][eInterfaceType],&ptr); |



3、初次分配后，会进行memset

|  |
| --- |
| psBbuIo->sFrontHaulTxBbuIoBufCtrl[j][i].sBufferList.pBuffers[k].pData = (uint8\_t \*)ptr; // k是符号索引  u32dptr = (uint32\_t\*)(ptr);  memset(u32dptr, 0x00, FPGA\_ALIGN(2\*273\*12\*4) + FH\_HEADER\_SIZE); |

其中：memset的长度为 2根天线iq数据长度+ 32byte(header);

## FH TX buffer填写

FH TX的实现函数在phy\_fpga\_dl\_set\_remap\_buffer中：

先填写Header，填写如下：

pktLen 是header+2个天线的iq数据的长度；

pktType = FTHTX\_OUT\_PKT\_TYPE //0x3

然后填写数据，当前函数只是将psPHYStateTx->remapSymbol[iPort][iSymb] 指针指向之前分配的sFrontHaulTxBbuIoBufCtrl中的pdata中，具体填写数据在remap的处理中，

|  |
| --- |
| for (iSymb = 0; iSymb < NUM\_OF\_SYMBOL\_PER\_SLOT; iSymb++)  {  pChar = psBbuIo->sFrontHaulTxBbuIoBufCtrl[bufIdx][nCellIdx].sBufferList.pBuffers[iSymb].pData;    /\*FH Tx header填写\*/  psHeader = (FHHeaderStruct \*)pChar;  psHeader->pktType = FTHTX\_OUT\_PKT\_TYPE;  psHeader->pktLen = nPktLen;  psHeader->symIdx = iSymb;  psHeader->frameIdx= nSlotAdvIdx /10 / pConfigParams->nNumOfSlotPerSubframe;  psHeader->subfIdx = ((nSlotAdvIdx % (FH\_MAX\_SUBFRAME\*pConfigParams->nNumOfSlotPerSubframe)) / pConfigParams->nNumOfSlotPerSubframe);  psHeader->slotIdx = nSlotAdvIdx % pConfigParams->nNumOfSlotPerSubframe;  for(iPort = 0; iPort < nPort; iPort++)  {  pDataChar[iPort] = pChar + FH\_HEADER\_SIZE + iPort\*oneSymDataSize;  psPHYStateTx->remapSymbol[iPort][iSymb] = (int16\_t \*)pDataChar[iPort]; // memset 在 dl post 函数中处理  }  } |

## FH TX memset的处理

下行FH TX填写数据之后，等发送完成即dl\_post之后，会进行memset处理，清空buffer，实现在函数 phy\_gnb\_bbupool\_post\_task\_dl\_reset\_buf 进行：

清除buffer内容的实现机制如下：

bufIdx = (nSlotAdvIdx+FH\_NUMBER\_OF\_BUFFER+RE\_RESET\_ADVANCE)%FH\_NUMBER\_OF\_BUFFER

// FH\_NUMBER\_OF\_BUFFER = 8 RE\_RESET\_ADVANCE= 3

例如：当前 nSlotAdvIdx = 2， 分配数据的时候，bufIdx= 2， 此时清除的bufIdx= (2+8+3)%8=5的buffer内容，也就是清除之前第5个bufIdx的数据；

|  |
| --- |
| //reset the Tx buffer  if (pPhyDiCtx->PhyStartMode == DI\_RADIO)  {  for (iSymb = 0; iSymb < NUM\_OF\_SYMBOL\_PER\_SLOT; iSymb++)  {  pChar = psBbuIo->sFrontHaulTxBbuIoBufCtrl[bufIdx][nCellIdx].sBufferList.pBuffers[iSymb].pData + FH\_HEADER\_SIZE;  if(NULL == pChar)  zLog(PHY\_LOG\_ERROR, "phy\_gnb\_bbupool\_post\_task\_dl\_reset\_buf, pChar = NULL, oneSymDataSize[%d]!!!!\n", oneSymDataSize);  memset((void\*)pChar, 0x0, 2\*oneSymDataSize);  }  } |

## 需要注意的问题

### 发送到FPGA的数据是否进行32byte对齐？

目前计算 一个符号的频域数据长度 是进行了32byte对齐的，按照如下方式：

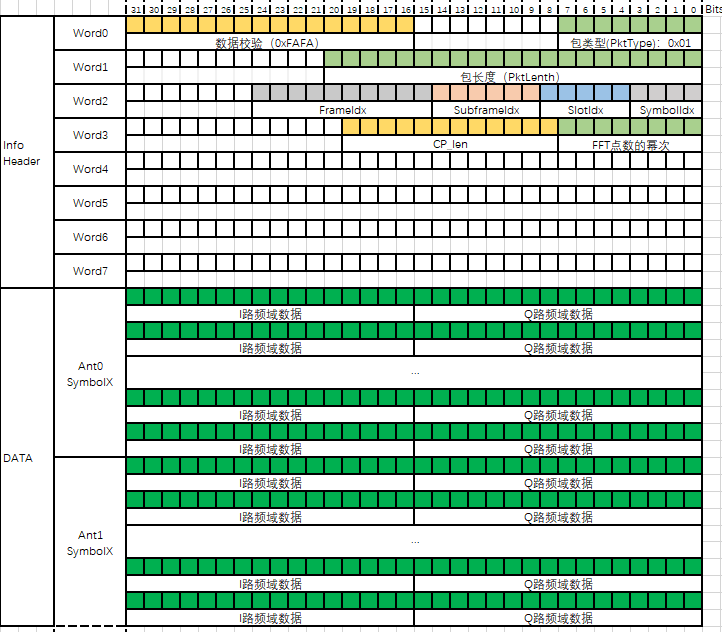
oneSymDataSize = FPGA\_ALIGN(pConfigParams->nNumOfPrbInFullBand \* 12 \* 4) //单位byte

### sFrontHaulTxBbuIoBufCtrl[bufIdx][nCellIdx].sBufferList.pBuffers[i]中的i使用symbol是否合理？

根据我们具体的设计将pBuffers修改为symbol索引，原来的设计是iPort索引，修改后需要修改内存分配；

# FH RX处理设计

FPGA FH过来的数据格式如下：

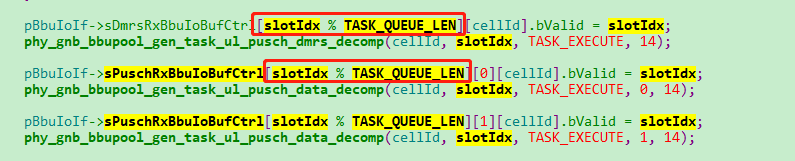


FPGA收齐了一个slot的数据后，在当前slot发送到X86侧， 数据内容是：1个symbol包括一个FH header，2个天线的iq数据，如下图所示：

|  |  |
| --- | --- |
| FH header 32Byte | |
| Symbol0 | ANT 0 IQ |
| ANT 1 IQ |
| FH header 32Byte | |
| Symbol1 | ANT 0 IQ |
| ANT 1 IQ |
| … … | |
| FH header 32Byte | |
| Symbol13 | ANT 0 IQ |
| ANT 1 IQ |

## FH RX接收的回调函数cpa\_fh\_rx\_callback

cpa\_fh\_rx\_callback中的处理：



函数中的 pBbuIoIf->sPuschRxBbuIoBufCtrl和pBbuIoIf->sDmrsRxBbuIoBufCtrl 仅在测试代码中使用，正式流程实现中没有使用；

phy\_gnb\_bbupool\_gen\_task\_ul\_pusch\_dmrs\_decomp 通过调用bbupool\_task\_generate会触发DMRS任务UL\_L1\_PUSCH\_DMRS\_DECOMP 7进行dmrs数据的接收；

phy\_gnb\_bbupool\_gen\_task\_ul\_pusch\_data\_decomp函数调用了2次，分别触发了UL\_L1\_PUSCH\_SYM0\_DECOMP 8和UL\_L1\_PUSCH\_SYM7\_DECOMP 8两个任务，进行前半个slot和后半slot的处理；

FHRX接收任务的实现参看后面的分析。

## FH RX 内存分配

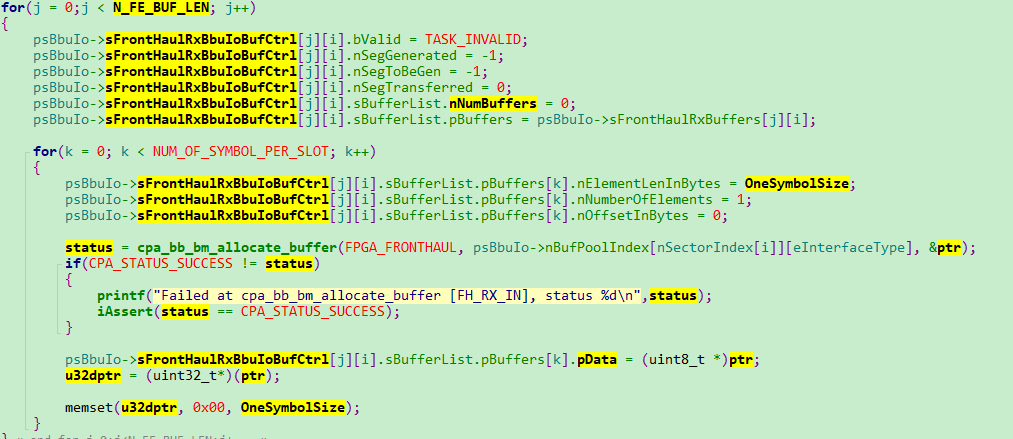
FH RX的内存分配在init\_fpga函数中进行处理：

1、先进行初始化，用来判断是否有足够内存可以分配：

|  |
| --- |
| cpa\_bb\_bm\_init (FPGA\_FRONTHAUL, &psBbuIo->nBufPoolIndex[nSectorIndex[i]][eInterfaceType],  N\_FE\_BUF\_LEN\*NUM\_OF\_SYMBOL\_PER\_SLOT, OneSymbolSize);  //OneSymbolSize = FPGA\_ALIGN(2\*273\*12\*4) + FH\_HEADER\_SIZE; |

2、通过函数cpa\_bb\_bm\_allocate\_buffer实现：

|  |
| --- |
| cpa\_bb\_bm\_allocate\_buffer(FPGA\_FRONTHAUL,psBbuIo->nBufPoolIndex[nSectorIndex[i]][eInterfaceType],&ptr); |



3、初次分配后，会进行memset

|  |
| --- |
| psBbuIo->sFrontHaulRxBbuIoBufCtrl [j][i].sBufferList.pBuffers[k].pData = (uint8\_t \*)ptr; // k是符号索引  u32dptr = (uint32\_t\*)(ptr);  memset(u32dptr, 0x00, OneSymbolSize); |

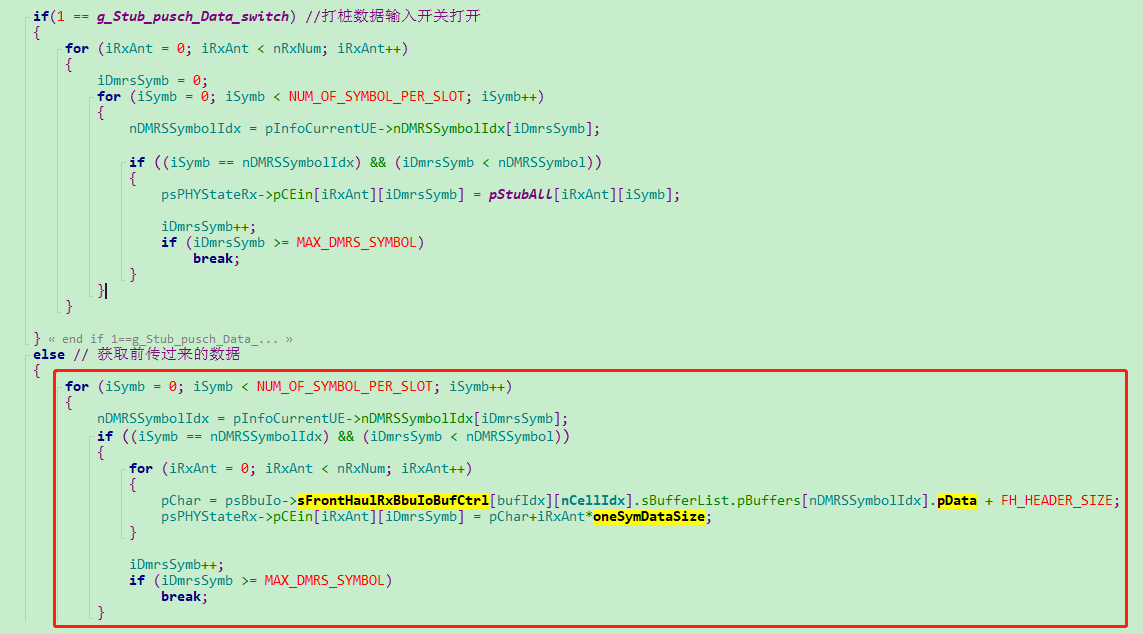
其中：memset的长度OneSymbolSize为 2根天线iq数据长度+ 32byte(header);

## FH RX 接收到的数据buffer填写到X86

### Pusch的处理



### DMRS的处理



### PUCCH(待补充)

### PRACH(待补充)

### SRS(待补充)