OpenAirInterface

MAC

Design Document Template

# Document Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Modified by | Date | Version | Comments |
| Paul | 2016/01/02 | 1.0 | RRM Introduction, Modules, Function block introduction |
| Paul | 2016/01/14 | 1.1 | Erase Finite State Machine |
| Vanessa | 2016/03/03 | 2.0 | (1) Modify Introduction  (2) Modify Modules  (3) Delete Function Blocks |
| Vanessa | 2016/03/04 | 2.1 | Add Fig. 1 |
| Vanessa | 2016/03/09 | 2.2 | (1) Add Fig. 2  (2) Add Fig. 3 |
| Vanessa | 2016/03/15 | 2.3 | Modify Function Flow of Each Module |
| Vanessa | 2016/03/29 | 2.4 | Modify Fig.2 |
| Vanessa | 2016/04/02 | 2.5 | (1) Modify Fig.2  (2) Modify Fig.3 |
| Eric | 2016/04/03 | 2.6 | Add Mudule/Process Table after System/Block Figures |
| Eric | 2016/06/08 | 2.7 | Modify Parameters of signal lines Table Format in System/Block level |
| Eric | 2016/07/10 | 3.0 | Draw left Process Figures and revise table numbers |
| Eric | 2016/11/10 | 3.1 | Modify DD Template base on Original DD and add Chapter 3 Channel, Chapter 4 Signal |
| Eric | 2016/11/16 | 3.2 | Modify Chapter Channel in DD Template and Modify Chapter 4 from Signal to Data Structure |
| Eric | 2016/12/05 | 3.3 | Finish Chapter 3 |

# Table of Content

[Document Revision History 2](#_Toc468707337)

[Table of Content 3](#_Toc468707338)

[1 Reference 4](#_Toc468707339)

[2 Introduction 5](#_Toc468707340)

[2.1 Definition and Abbreviations 5](#_Toc468707341)

[2.2 Overview 5](#_Toc468707342)

[3 Channel 7](#_Toc468707343)

[3.1 MAC-RRC 7](#_Toc468707344)

[3.1.1 UL\_CCCH 7](#_Toc468707345)

[3.1.2 DL\_BCCH 8](#_Toc468707346)

[3.1.3 RA\_CCCH 8](#_Toc468707347)

[3.1.4 Config\_BCCH 9](#_Toc468707348)

[3.1.5 Config\_CCCH 9](#_Toc468707349)

[3.1.6 Config\_DCCH 11](#_Toc468707350)

[3.2 MAC-RLC 12](#_Toc468707351)

[3.2.1 UL\_DCCH 12](#_Toc468707352)

[3.2.2 UL\_DTCH 12](#_Toc468707353)

[3.2.3 DL\_DCCH 13](#_Toc468707354)

[3.2.4 DL\_DTCH 13](#_Toc468707355)

[3.3 MAC-PHY 14](#_Toc468707356)

[3.3.1 TX\_DLSCH 14](#_Toc468707357)

[3.3.2 RX\_ULSCH 15](#_Toc468707358)

[3.3.3 TX\_DCI 17](#_Toc468707359)

[3.3.4 To\_MAC 17](#_Toc468707360)

[3.3.5 To\_PHY 19](#_Toc468707361)

[3.4 Inter Blocks 22](#_Toc468707362)

[3.4.1 PHY\_UL 22](#_Toc468707363)

[3.4.2 UL\_PHY 22](#_Toc468707364)

[3.4.3 PHY\_RX 23](#_Toc468707365)

[3.4.4 PHY\_DL 23](#_Toc468707366)

[3.4.5 DL\_PHY 24](#_Toc468707367)

[3.4.6 PHY\_SI 24](#_Toc468707368)

[3.4.7 SI\_PHY 25](#_Toc468707369)

[3.4.8 PHY\_RA 25](#_Toc468707370)

[3.4.9 RA\_PHY 26](#_Toc468707371)

[3.4.10 Config\_PHY 26](#_Toc468707372)

[4 Important Data Structure 27](#_Toc468707373)

[4.1 eNB\_MAC\_INST 27](#_Toc468707374)

[4.2 eNB common channels 28](#_Toc468707375)

[4.3 eNB \*RA\_template 29](#_Toc468707376)

[4.4 eNB DCI\_PDU\* 31](#_Toc468707377)

[4.5 eNB ULSCH\_PDU\* 31](#_Toc468707378)

[4.6 eNB DLSCH\_PDU\* 31](#_Toc468707379)

[4.7 eNB CCCH\_PDU\* 31](#_Toc468707380)

[4.8 eNB BCCH\_PDU\* 32](#_Toc468707381)

[4.9 UE list\* 32](#_Toc468707382)

[4.10 mac\_rlc\_status\_resp\_t 33](#_Toc468707383)

[5 Block 33](#_Toc468707384)

[5.1 PHY\_MAC\_Interface 34](#_Toc468707385)

[5.1.1 Introduction 34](#_Toc468707386)

[5.2 DLSCH\_Process 35](#_Toc468707387)

[5.2.1 Introduction 35](#_Toc468707388)

[5.3 ULSCH\_Process 37](#_Toc468707389)

[5.3.1 Introduction 37](#_Toc468707390)

[5.4 RA\_Process 39](#_Toc468707391)

[5.4.1 Introduction 39](#_Toc468707392)

[5.5 SI\_Process 41](#_Toc468707393)

[5.5.1 Introduction 41](#_Toc468707394)

[5.6 Configuration 43](#_Toc468707395)

[5.6.1 Introduction 43](#_Toc468707396)

# 1 Reference

|  |  |  |  |
| --- | --- | --- | --- |
| **Spec.** | **Release** | **Page** | **Related work** |
| 3GPP 36.321 | V8.6 |  |  |
| 3GPP 36.321 | V13.2 |  |  |

Table 1. Related work will be the chapter in this document.

# 2 Introduction

## 2.1 Definition and Abbreviations

|  |  |
| --- | --- |
| **Term** | **Definition** |
| RRM | Radio Resource Management |
| RRC | Radio Resource Control |
| PDCP | Packet Data Convergence Protocol |
| RLC | Radio Link Control |
| MAC | Media access control |
| PHY | Physical layer |

Table 2. Abbreviations and definition

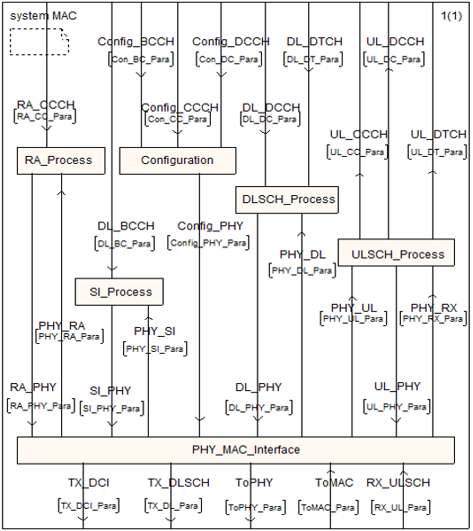
## 2.2 Overview

Fig. 1 shows the modules implemented in the MAC layer .

In this chapter, need to point out :

1. Modules within MAC layer

2. The upper or lower relationship between Modules

Fig.1 System level of MAC layer. 

# 3 Channel

## 3.1 MAC-RRC

### 3.1.1 UL\_CCCH

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Block** | **Primitives** | **Direction** | **Input Parameters** | **Output Parameters**  (UL\_CC\_Para) |
| ULSCH\_process | mac\_rrc\_data\_ind() | MAC→RRC | enb\_mod\_idP,  CC\_idP, frameP,subframeP, rntiP, CCCH, (uint8\_t\*)payload\_ptr, rx\_lengths[i], ENB\_FLAG\_YES, enb\_mod\_idP,  0 | (uint8\_t\*)payload\_ptr, rx\_lengths[i], |

Table . Parameters of Cross Layer Channel.

### 3.1.2 DL\_BCCH

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Block** | **Primitives** | **Direction** | **Input Parameters** | **Output Parameters**  (DL\_BC\_Para) |
| SI\_Process | mac\_rrc\_data\_req() | RRC→MAC | module\_idP  CC\_id  frameP  BCCH  1  &eNB->common\_channels[CC\_id].BCCH\_pdu.payload[0],  1  module\_idP | rrc\_sdu\_length  BCCH\_pdu.payload[0] |

Tablex . Parameters of Cross Layer Channel.

### 3.1.3 RA\_CCCH

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Block** | **Primitives** | **Direction** | **Input Parameters** | **Output Parameters**  (RA\_CC\_Para) |
| RA\_Process | mac\_rrc\_data\_req() | RRC→MAC | module\_idP,  CC\_id,  frameP,  CCCH,  1,  &eNB->common\_channels[CC\_id].CCCH\_pdu.payload[0],  ENB\_FLAG\_YES,  module\_idP | rrc\_sdu\_length,  CCCH\_pdu.payload[0] |

Tablex . Parameters of Cross Layer Channel.

### 3.1.4 Config\_BCCH

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Block** | **Primitives** | **Direction** | **Input Parameters** | **Output Parameters** |
| Configuration | mac\_rrc\_config\_req() | RRC→MAC | Mod\_id  CC\_id  eNB\_flagP  \*RadioResourceConfigCommonSIB-NB  \*SIwindowsize  \*SIperiod  \*ul\_CarrierFreq  \*additionalSpectrumEmission  (rntiP)NULL  (eNB\_index) NULL  (PhysicalConfigDedicated-NB) NULL  (\*MAC-MainConfig-NB) NULL  (logicalChannelIdentity) NULL  (\*LogicalChannelConfig-NB) NULL | N/A |

Table . Parameters of Cross Layer Channel.

### 3.1.5 Config\_CCCH

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Block** | **Primitives** | **Direction** | **Input Parameters** | **Output Parameters** |
| Configuration | mac\_rrc\_config\_req() | RRC→MAC | Mod\_id  CC\_id  eNB\_flagP  rntiP  PhysicalConfigDedicated-NB  \*MAC-MainConfig-NB  logicalChannelIdentity  \*LogicalChannelConfig-NB  (eNB\_index ) NULL  (\*RadioResourceConfigCommonSIB-NB) NULL  (\*SIwindowsize) NULL  (\*SIperiod) NULL  (\*ul\_CarrierFreq) NULL  (\*additionalSpectrumEmission) NULL | N/A |

Table . Parameters of Cross Layer Channel.

### 3.1.6 Config\_DCCH

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Block** | **Primitives** | **Direction** | **Input Parameters** | **Output Parameters** |
| Configuration | mac\_rrc\_config\_req() | RRC→MAC | Mod\_id  CC\_id  eNB\_flagP  rntiP  PhysicalConfigDedicated-NB  \*MAC-MainConfig-NB  logicalChannelIdentity  \*LogicalChannelConfig-NB  (eNB\_index ) NULL  (\*RadioResourceConfigCommonSIB-NB) NULL  (\*SIwindowsize) NULL  (\*SIperiod) NULL  (\*ul\_CarrierFreq) NULL  (\*additionalSpectrumEmission) NULL | N/A |

## 3.2 MAC-RLC

### 3.2.1 UL\_DCCH

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Block** | **Primitives** | **Direction** | **Input Parameters**  (UL\_DC\_Para) | **Output Parameters** |
| ULSCH\_Process | mac\_rlc\_data\_ind() | MAC→RLC | enb\_mod\_idP, rntiP, enb\_mod\_idP,  frameP, ENB\_FLAG\_YES,MBMS\_FLAG\_NO, rx\_lcids[i],  (char\*)payload\_ptr,  rx\_lengths[i],  1, NULL | (char\*)payload\_ptr,  rx\_lengths[i], |

Tablex . Parameters of Cross Layer Channel.

### 3.2.2 UL\_DTCH

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Block** | **Primitives** | **Direction** | **Input Parameters**  (UL\_DT\_Para) | **Output Parameters** |
| ULSCH\_Process | mac\_rlc\_data\_ind() | MAC→RLC | enb\_mod\_idP, rntiP, enb\_mod\_idP,  frameP, ENB\_FLAG\_YES,MBMS\_FLAG\_NO, rx\_lcids[i],  (char\*)payload\_ptr,  rx\_lengths[i],  1, NULL | (char\*)payload\_ptr,  rx\_lengths[i], |

Tablex . Parameters of Cross Layer Channel.

### 3.2.3 DL\_DCCH

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Block** | **Primitives** | **Direction** | **Input Parameters** | **Output Parameters**  (DL\_DC\_Para) |
| DLSCH\_Process | mac\_rlc\_data\_req() | MAC→RLC | module\_idP, rntiP, eNB\_index, frameP, enb\_flagP, channel\_idP, buffer\_pP | ret\_tb\_size  Put RLC data from different mode to MAC dlsch\_buffer |
| mac\_rlc\_serialize\_tb() | RLC→MAC | buffer\_pP, transport\_blocksP | ret\_tb\_size  Put RLC data to MAC dlsch\_buffer  //Used by mac\_rlc\_data\_req() |
| mac\_rlc\_status\_ind | MAC→RLC | module\_idP,  rntiP,  eNB\_index,  frameP,  enb\_flagP,  channel\_idP,  tb\_sizeP | mac\_rlc\_status\_resp  a structure have data information which for scheduled |

Tablex . Parameters of Cross Layer Channel.

### 3.2.4 DL\_DTCH

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Block** | **Primitives** | **Direction** | **Input Parameters** | **Output Parameters**  (DL\_DT\_Para) |
| DLSCH\_Process | mac\_rlc\_data\_req() | MAC→RLC | module\_idP, rntiP, eNB\_index, frameP, enb\_flagP, channel\_idP, buffer\_pP | ret\_tb\_size  Put RLC data from different mode to MAC dlsch\_buffer |
| mac\_rlc\_serialize\_tb() | RLC→MAC | buffer\_pP, transport\_blocksP | ret\_tb\_size  Put RLC data to MAC dlsch\_buffer  //Used by mac\_rlc\_data\_req() |
| mac\_rlc\_status\_ind | MAC→RLC | module\_idP,  rntiP,  eNB\_index,  frameP,  enb\_flagP,  channel\_idP,  tb\_sizeP | mac\_rlc\_status\_resp  a structure have data information which for scheduled |

Tablex . Parameters of Cross Layer Channel.

## 3.3 MAC-PHY

### 3.3.1 TX\_DLSCH

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Block** | **Primitives** | **Direction** | **Input Parameters** | **Output Parameters**  (TX\_DL\_Para) |
| PHY\_MAC\_Interface | get\_dlsch\_sdu() | PHY→MAC | phy\_vars\_eNB->Mod\_id,  phy\_vars\_eNB->CC\_id,  phy\_vars\_eNB->proc[sched\_subframe].frame\_tx,  phy\_vars\_eNB->dlsch\_eNB[(uint8\_t)UE\_id][0]->rnti,  0 | (unsigned char \*)&eNB->UE\_list.DLSCH\_pdu[CC\_id][TBindex][UE\_id].payload[0] |
| get\_dlsch\_sdu() | PHY→MAC | phy\_vars\_eNB->Mod\_id, phy\_vars\_eNB->CC\_id, phy\_vars\_eNB->proc[sched\_subframe].frame\_tx, SI\_RNTI, 0 | (unsigned char \*)&eNB->common\_channels[CC\_id].BCCH\_pdu.payload[0] |

Tablex . Parameters of Cross Layer Channel.

### 3.3.2 RX\_ULSCH

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Block** | **Primitives** | **Direction** | **Input Parameters**  (RX\_UL\_Para) | **Output Parameters**  (PHY\_RX) |
| PHY\_MAC\_Interface | rx\_sdu() | PHY→MAC | phy\_vars\_eNB->Mod\_id, phy\_vars\_eNB->CC\_id, frame,  subframe, phy\_vars\_eNB->ulsch\_eNB[i]->rnti, NULL, 0, harq\_pid, &phy\_vars\_eNB->ulsch\_eNB[i]->Msg3\_flag | phy\_vars\_eNB->Mod\_id, phy\_vars\_eNB->CC\_id, frame,  subframe, phy\_vars\_eNB->ulsch\_eNB[i]->rnti, NULL, 0, harq\_pid, &phy\_vars\_eNB->ulsch\_eNB[i]->Msg3\_flag |
| rx\_sdu() | PHY→MAC | phy\_vars\_eNB->Mod\_id, phy\_vars\_eNB->CC\_id, frame,  subframe,  phy\_vars\_eNB->ulsch\_eNB[i]->rnti, phy\_vars\_eNB->ulsch\_eNB[i]->harq\_processes[harq\_pid]->b, phy\_vars\_eNB->ulsch\_eNB[i]->harq\_processes[harq\_pid]->TBS>>3, harq\_pid, &phy\_vars\_eNB->ulsch\_eNB[i]->Msg3\_flag | phy\_vars\_eNB->Mod\_id, phy\_vars\_eNB->CC\_id, frame,  subframe,  phy\_vars\_eNB->ulsch\_eNB[i]->rnti, phy\_vars\_eNB->ulsch\_eNB[i]->harq\_processes[harq\_pid]->b, phy\_vars\_eNB->ulsch\_eNB[i]->harq\_processes[harq\_pid]->TBS>>3, harq\_pid, &phy\_vars\_eNB->ulsch\_eNB[i]->Msg3\_flag |
| rx\_sdu() | PHY→MAC | phy\_vars\_eNB->Mod\_id, phy\_vars\_eNB->CC\_id, frame,  subframe, phy\_vars\_eNB->ulsch\_eNB[i]->rnti, phy\_vars\_eNB->ulsch\_eNB[i]->harq\_processes[harq\_pid]->b, phy\_vars\_eNB->ulsch\_eNB[i]->harq\_processes[harq\_pid]->TBS>>3, harq\_pid, NULL | phy\_vars\_eNB->Mod\_id, phy\_vars\_eNB->CC\_id, frame,  subframe, phy\_vars\_eNB->ulsch\_eNB[i]->rnti, phy\_vars\_eNB->ulsch\_eNB[i]->harq\_processes[harq\_pid]->b, phy\_vars\_eNB->ulsch\_eNB[i]->harq\_processes[harq\_pid]->TBS>>3, harq\_pid, NULL |

Tablex . Parameters of Cross Layer Channel.

### 3.3.3 TX\_DCI

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Block** | **Primitives** | **Direction** | **Input Parameters** | **Output Parameters**  (TX\_DCI\_Para) |
| PHY\_MAC\_Interface | get\_dci\_sdu() | PHY→MAC | phy\_vars\_eNB->Mod\_id, phy\_vars\_eNB->CC\_id, phy\_vars\_eNB->proc[sched\_subframe].frame\_tx, subframe | &eNB\_mac\_inst[module\_idP].common\_channels[CC\_id].DCI\_pdu |

Tablex . Parameters of Cross Layer Channel.

### 3.3.4 To\_MAC

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Block** | **Primitives** | **Direction** | **Input Parameters**  (PHY\_Para) | **Output Parameters** |
| PHY\_MAC\_Interface | eNB\_dlsch\_ulsch\_scheduler() | PHY→MAC | phy\_vars\_eNB->Mod\_id,  0,  phy\_vars\_eNB->proc[sched\_subframe].frame\_tx,  subframe | N/A |
| initiate\_ra\_proc() | PHY→MAC | phy\_vars\_eNB->Mod\_id, phy\_vars\_eNB->CC\_id, frame, preamble\_max, preamble\_delay\_list[preamble\_max]\*update\_TA, 0,  subframe,0 | (RA\_TEMPLATE \*)&eNB\_mac\_inst[module\_idP].common\_channels[CC\_id].RA\_template[0] |
| cancel\_ra\_proc | PHY→MAC | phy\_vars\_eNB->Mod\_id, phy\_vars\_eNB->CC\_id, phy\_vars\_eNB->proc[sched\_subframe].frame\_tx, crnti | (RA\_TEMPLATE \*)&eNB\_mac\_inst[module\_idP].common\_channels[CC\_id].RA\_template[0] |
|  | cancel\_ra\_proc | PHY→MAC | phy\_vars\_eNB->Mod\_id, phy\_vars\_eNB->CC\_id, frame, phy\_vars\_eNB->eNB\_UE\_stats[i].crnti | (RA\_TEMPLATE \*)&eNB\_mac\_inst[module\_idP].common\_channels[CC\_id].RA\_template[0] |
|  | cancel\_ra\_proc | PHY→MAC | phy\_vars\_eNB->Mod\_id, phy\_vars\_eNB->CC\_id, frame, phy\_vars\_eNB->eNB\_UE\_stats[i].crnti | (RA\_TEMPLATE \*)&eNB\_mac\_inst[module\_idP].common\_channels[CC\_id].RA\_template[0] |
|  | fill\_rar() | PHY→MAC | phy\_vars\_eNB->Mod\_id, phy\_vars\_eNB->CC\_id, phy\_vars\_eNB->proc[sched\_subframe].frame\_tx, dlsch\_input\_buffer, phy\_vars\_eNB->lte\_frame\_parms.N\_RB\_UL, input\_buffer\_length | eNB\_mac\_inst[module\_idP].common\_channels[CC\_id].RA\_template[ra\_idx].rnti,  \*rar |

Tablex . Parameters of Cross Layer Channel.

### 3.3.5 To\_PHY

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Block** | **Primitives** | **Direction** | **Input Parameters**  (PHY\_Para) | **Output Parameters** |
| PHY\_MAC\_Interface | phy\_config\_dedicated\_eNB() | MAC→PHY | Mod\_id,  CC\_id, UE\_RNTI(Mod\_id, UE\_id), physicalConfigDedicated | phy\_vars\_eNB->physicalConfigDedicated[UE\_id] |
| phy\_config\_sib2\_eNB() | MAC→PHY | Mod\_id,  CC\_id, radioResourceConfigCommon,  ul\_CarrierFreq, ul\_Bandwidth, additionalSpectrumEmission, mbsfn\_SubframeConfigList | &PHY\_vars\_eNB\_g[Mod\_id][CC\_id]->lte\_frame\_parms |
| get\_ue\_active\_harq\_pid() | MAC→PHY | module\_idP,  CC\_id,  RA\_template->rnti,  frameP,  subframeP,  &harq\_pid,  &round,  0 | \*harq\_pid  \*round |
| get\_ue\_active\_harq\_pid() | MAC→PHY | module\_idP,CC\_id,rnti, frameP,  subframeP, &ue\_sched\_ctl->harq\_pid[CC\_id],&ue\_sched\_ctl->round[CC\_id], 0 | \*harq\_pid  \*round |
| get\_ue\_active\_harq\_pid() | MAC→PHY | module\_idP,  CC\_id,  rnti,  frameP,  subframeP,  &harq\_pid,  &round,  1 | \*harq\_pid  \*round |
| get\_ue\_active\_harq\_pid() | MAC→PHY | module\_idP,  CC\_id,  rnti,  frameP,  subframeP,  &harq\_pid,  &round,  1 | \*harq\_pid  \*round |
| get\_TBS\_DL() | MAC→PHY | mcs,  nb\_rb | TBS |
| get\_TBS\_UL() | MAC→PHY | mcs,  rb\_table[rb\_table\_index] | TBS |
| get\_lte\_frame\_parms() | MAC→PHY | module\_idP,  CC\_id | &PHY\_vars\_eNB\_g[Mod\_id][CC\_id]->lte\_frame\_parms |
| get\_rballoc() | MAC→PHY | (DCI1A\_1\_5MHz\_TDD\_1\_6\_t\*)&RA\_template->RA\_alloc\_pdu1[0])->vrb\_type, ((DCI1A\_1\_5MHz\_TDD\_1\_6\_t\*)&RA\_template->RA\_alloc\_pdu1[0])->rballoc | ((DCI1A\_1\_5MHz\_TDD\_1\_6\_t\*)&RA\_template->RA\_alloc\_pdu1[0])->rballoc is from  computeRIV() |
| get\_rballoc() | MAC→PHY | 0,  ((DCI1A\_1\_5MHz\_TDD\_1\_6\_t\*)BCCH\_alloc\_pdu)->rballoc | ((DCI1A\_1\_5MHz\_TDD\_1\_6\_t\*)BCCH\_alloc\_pdu)->rballoc is from  computeRIV() |
| computeRIV() | MAC→PHY | PHY\_vars\_eNB\_g[module\_idP][CC\_id]->lte\_frame\_parms.N\_RB\_DL,  first\_rb,  4 | RIV |
| …. | …. | … | … | … |

## 3.4 Inter Blocks

### 3.4.1 PHY\_UL

|  |  |  |  |
| --- | --- | --- | --- |
| **From Block** | **To Block** | **Channel Parameters**  **(**PHY\_UL\_Para**)** | **Note** |
| PHY\_MAC\_Interface | ULSCH\_Process | module\_idP  frameP  cooperation\_flag  subframeP  sched\_subframe | The input parameter of  function schedule\_ulsch(),  Sched\_subframe = subframeP+4 |

Tablex . Parameters of Inter Blocks Channel.

### 3.4.2 UL\_PHY

|  |  |  |  |
| --- | --- | --- | --- |
| **From Block** | **To Block** | **Channel Parameters**  **(**UL\_PHY\_Para**)** | **Note** |
| ULSCH\_Process | PHY\_MAC\_Interface | &eNB\_mac\_inst[module\_idP].common\_channels[CC\_id].DCI\_pdu | Return the position of DCI\_pdu |

Tablex . Parameters of Inter Blocks Channel.

### 3.4.3 PHY\_RX

|  |  |  |  |
| --- | --- | --- | --- |
| **From Block** | **To Block** | **Channel Parameters**  **(**PHY\_RX\_Para**)** | **Note** |
| PHY\_MAC\_Interface | ULSCH\_Process | phy\_vars\_eNB->Mod\_id, phy\_vars\_eNB->CC\_id, frame,subframe,  phy\_vars\_eNB->ulsch\_eNB[i]->rnti, phy\_vars\_eNB->ulsch\_eNB[i]->harq\_processes[harq\_pid]->b,  phy\_vars\_eNB->ulsch\_eNB[i]->harq\_processes[harq\_pid]->TBS>>3, harq\_pid, NULL | This function describe how MAC received data from PHY |

Tablex . Parameters of Inter Blocks Channel.

### 3.4.4 PHY\_DL

|  |  |  |  |
| --- | --- | --- | --- |
| **From Block** | **To Block** | **Channel Parameters**  **(**DL\_PHY\_Para**)** | **Note** |
| DLSCH\_Process | PHY\_MAC\_Interface | module\_idP,  frameP,  subframeP,  \*nb\_rb\_used0,  \*nCCE\_used,  mbsfn\_flag | The input parameter of  function schedule\_ue\_spec() |

Tablex . Parameters of Inter Blocks Channel.

### 3.4.5 DL\_PHY

|  |  |  |  |
| --- | --- | --- | --- |
| **From Block** | **To Block** | **Channel Parameters**  **(**PHY\_DL\_Para**)** | **Note** |
| DLSCH\_Process | PHY\_MAC\_Interface | (unsigned char \*)&eNB->UE\_list.DLSCH\_pdu[CC\_id][TBindex][UE\_id].payload[0],  &eNB\_mac\_inst[module\_idP].common\_channels[CC\_id].DCI\_pdu | MAC PDU to PHY which  from RLC logical channel  Return the DCI PDU |

Tablex . Parameters of Inter Blocks Channel.

### 3.4.6 PHY\_SI

|  |  |  |  |
| --- | --- | --- | --- |
| **From Block** | **To Block** | **Channel Parameters**  **(**PHY\_SI\_Para**)** | **Note** |
| PHY\_MAC\_Interface | SI\_Process | module\_idP  frameP  nprbP  nCCEP | The input parameter of  function schedule\_SI() |

Tablex . Parameters of Inter Blocks Channel.

### 3.4.7 SI\_PHY

|  |  |  |  |
| --- | --- | --- | --- |
| **From Block** | **To Block** | **Channel Parameters**  **(**SI\_PHY\_Para**)** | **Note** |
| SI\_Process | PHY\_MAC\_Interface | BCCH\_pdu.payload[0] | Return the position of BCCH\_pdu.padload |

Tablex . Parameters of Inter Blocks Channel.

### 3.4.8 PHY\_RA

|  |  |  |  |
| --- | --- | --- | --- |
| **From Block** | **To Block** | **Channel Parameters**  **(**PHY\_RA\_Para**)** | **Note** |
| PHY\_MAC\_Interface | RA\_Process | RA\_template->RA\_active | Flag to indicate this process is active. |
| RA\_template->generate\_rar | Flag to indicate the eNB should generate RAR. This is triggered by detection of PRACH. |
| RA\_template->generate\_Msg4 | Flag to indicate the eNB should generate Msg4 upon reception of SDU from RRC. This is triggered by first ULSCH reception at eNB for new user. |

Tablex . Parameters of Inter Blocks Channel.

### 3.4.9 RA\_PHY

|  |  |  |  |
| --- | --- | --- | --- |
| **From Block** | **To Block** | **Channel Parameters**  **(**RA\_PHY\_Para**)** | **Note** |
| RA\_Process | PHY\_MAC\_Interface | RA\_template->RA\_alloc\_pdu2[0] | Actual DCI to transmit for Msg4/ContRes. |

Tablex . Parameters of Inter Blocks Channel.

### 3.4.10 Config\_PHY

|  |  |  |  |
| --- | --- | --- | --- |
| **From Block** | **To Block** | **Channel Parameters**  **(**Config\_PHY\_Para**)** | **Note** |
| Configuration | PHY\_MAC\_Interface | Mod\_id  CC\_id  rntiP  PhysicalConfigDedicated-NB | The Channel Parameters is the input of  phy\_config\_dedicated\_eNB() |

Tablex . Parameters of Cross Layer Channel.

# 4 Important Data Structure

The following data structure define in LAYER2/MAC/defs.h

## 4.1 eNB\_MAC\_INST

/\*! \brief top level eNB MAC structure \*/

typedef struct {

///

uint16\_t Node\_id;

/// frame counter

frame\_t frame;

/// subframe counter

sub\_frame\_t subframe;

/// Common cell resources

COMMON\_channels\_t common\_channels[MAX\_NUM\_CCs];

UE\_list\_t UE\_list;

///subband bitmap configuration

SBMAP\_CONF sbmap\_conf;

/// CCE table used to build DCI scheduling information

int CCE\_table[MAX\_NUM\_CCs][800];

/// active flag for Other lcid

uint8\_t lcid\_active[NB\_RB\_MAX];

/// eNB stats

eNB\_STATS eNB\_stats[MAX\_NUM\_CCs];

// MAC function execution peformance profiler

/// processing time of eNB scheduler

time\_stats\_t eNB\_scheduler;

/// processing time of eNB scheduler for SI

time\_stats\_t schedule\_si;

/// processing time of eNB scheduler for Random access

time\_stats\_t schedule\_ra;

/// processing time of eNB ULSCH scheduler

time\_stats\_t schedule\_ulsch;

/// processing time of eNB DCI generation

time\_stats\_t fill\_DLSCH\_dci;

/// processing time of eNB MAC preprocessor

time\_stats\_t schedule\_dlsch\_preprocessor;

/// processing time of eNB DLSCH scheduler

time\_stats\_t schedule\_dlsch; // include rlc\_data\_req + MAC header + preprocessor

/// processing time of eNB MCH scheduler

time\_stats\_t schedule\_mch;

/// processing time of eNB ULSCH reception

time\_stats\_t rx\_ulsch\_sdu; // include rlc\_data\_ind

} eNB\_MAC\_INST;

## 4.2 eNB common channels

/\*! \brief eNB common channels \*/

typedef struct {

/// Outgoing DCI for PHY generated by eNB scheduler

DCI\_PDU DCI\_pdu;

/// Outgoing BCCH pdu for PHY

BCCH\_PDU BCCH\_pdu;

/// Outgoing BCCH DCI allocation

uint32\_t BCCH\_alloc\_pdu;

/// Outgoing CCCH pdu for PHY

CCCH\_PDU CCCH\_pdu;

RA\_TEMPLATE RA\_template[NB\_RA\_PROC\_MAX];

/// VRB map for common channels

uint8\_t vrb\_map[100];

/// MBSFN SubframeConfig

struct MBSFN\_SubframeConfig \*mbsfn\_SubframeConfig[8];

/// number of subframe allocation pattern available for MBSFN sync area

uint8\_t num\_sf\_allocation\_pattern;

#ifdef Rel10

/// MBMS Flag

uint8\_t MBMS\_flag;

/// Outgoing MCCH pdu for PHY

MCCH\_PDU MCCH\_pdu;

/// MCCH active flag

uint8\_t msi\_active;

/// MCCH active flag

uint8\_t mcch\_active;

/// MTCH active flag

uint8\_t mtch\_active;

/// number of active MBSFN area

uint8\_t num\_active\_mbsfn\_area;

/// MBSFN Area Info

struct MBSFN\_AreaInfo\_r9 \*mbsfn\_AreaInfo[MAX\_MBSFN\_AREA];

/// PMCH Config

struct PMCH\_Config\_r9 \*pmch\_Config[MAX\_PMCH\_perMBSFN];

/// MBMS session info list

struct MBMS\_SessionInfoList\_r9 \*mbms\_SessionList[MAX\_PMCH\_perMBSFN];

/// Outgoing MCH pdu for PHY

MCH\_PDU MCH\_pdu;

#endif

#ifdef CBA

/// number of CBA groups

uint8\_t num\_active\_cba\_groups;

/// RNTI for each CBA group

uint16\_t cba\_rnti[NUM\_MAX\_CBA\_GROUP];

/// MCS for each CBA group

uint8\_t group\_mcs[NUM\_MAX\_CBA\_GROUP];

#endif

} COMMON\_channels\_t;

## 4.3 eNB \*RA\_template

/\*! \brief eNB template for the Random access information \*/

typedef struct {

/// Flag to indicate this process is active

boolean\_t RA\_active;

/// Size of DCI for RA-Response (bytes)

uint8\_t RA\_dci\_size\_bytes1;

/// Size of DCI for RA-Response (bits)

uint8\_t RA\_dci\_size\_bits1;

/// Actual DCI to transmit for RA-Response

uint8\_t RA\_alloc\_pdu1[(MAX\_DCI\_SIZE\_BITS>>3)+1];

/// DCI format for RA-Response (should be 1A)

uint8\_t RA\_dci\_fmt1;

/// Size of DCI for Msg4/ContRes (bytes)

uint8\_t RA\_dci\_size\_bytes2;

/// Size of DCI for Msg4/ContRes (bits)

uint8\_t RA\_dci\_size\_bits2;

/// Actual DCI to transmit for Msg4/ContRes

uint8\_t RA\_alloc\_pdu2[(MAX\_DCI\_SIZE\_BITS>>3)+1];

/// DCI format for Msg4/ContRes (should be 1A)

uint8\_t RA\_dci\_fmt2;

/// Flag to indicate the eNB should generate RAR. This is triggered by detection of PRACH

uint8\_t generate\_rar;

/// Subframe where preamble was received

uint8\_t preamble\_subframe;

/// Subframe where Msg3 is to be sent

uint8\_t Msg3\_subframe;

/// Flag to indicate the eNB should generate Msg4 upon reception of SDU from RRC. This is triggered by first ULSCH reception at eNB for new user.

uint8\_t generate\_Msg4;

/// Flag to indicate that eNB is waiting for ACK that UE has received Msg3.

uint8\_t wait\_ack\_Msg4;

/// UE RNTI allocated during RAR

rnti\_t rnti;

/// RA RNTI allocated from received PRACH

uint16\_t RA\_rnti;

/// Received preamble\_index

uint8\_t preamble\_index;

/// Received UE Contention Resolution Identifier

uint8\_t cont\_res\_id[6];

/// Timing offset indicated by PHY

int16\_t timing\_offset;

/// Timeout for RRC connection

int16\_t RRC\_timer;

} RA\_TEMPLATE;

## 4.4 eNB DCI\_PDU\*

/\*!\brief DCI PDU filled by MAC for the PHY \*/

typedef struct {

uint8\_t Num\_ue\_spec\_dci ;

uint8\_t Num\_common\_dci ;

// uint32\_t nCCE;

uint32\_t num\_pdcch\_symbols;

DCI\_ALLOC\_t dci\_alloc[NUM\_DCI\_MAX] ;

} DCI\_PDU;

## 4.5 eNB ULSCH\_PDU\*

/\*! \brief Uplink SCH PDU Structure \*/

typedef struct {

int8\_t payload[SCH\_PAYLOAD\_SIZE\_MAX]; /\*!< \brief SACH payload \*/

uint16\_t Pdu\_size;

} \_\_attribute\_\_ ((\_\_packed\_\_)) ULSCH\_PDU;

## 4.6 eNB DLSCH\_PDU\*

/\*! \brief Downlink SCH PDU Structure \*/

typedef struct {

int8\_t payload[8][SCH\_PAYLOAD\_SIZE\_MAX];

uint16\_t Pdu\_size[8];

} \_\_attribute\_\_ ((\_\_packed\_\_)) DLSCH\_PDU;

## 4.7 eNB CCCH\_PDU\*

/\*! \brief CCCH payload \*/

typedef struct {

uint8\_t payload[CCCH\_PAYLOAD\_SIZE\_MAX] ;

} \_\_attribute\_\_((\_\_packed\_\_))CCCH\_PDU;

## 4.8 eNB BCCH\_PDU\*

/\*! \brief BCCH payload \*/

typedef struct {

uint8\_t payload[BCCH\_PAYLOAD\_SIZE\_MAX] ;

} \_\_attribute\_\_((\_\_packed\_\_))BCCH\_PDU;

## 4.9 UE list\*

/\*! \brief UE list used by eNB to order UEs/CC for scheduling\*/

typedef struct {

/// DLSCH pdu

DLSCH\_PDU DLSCH\_pdu[MAX\_NUM\_CCs][2][NUMBER\_OF\_UE\_MAX];

/// DCI template and MAC connection parameters for UEs

UE\_TEMPLATE UE\_template[MAX\_NUM\_CCs][NUMBER\_OF\_UE\_MAX];

/// DCI template and MAC connection for RA processes

int pCC\_id[NUMBER\_OF\_UE\_MAX];

/// sorted downlink component carrier for the scheduler

int ordered\_CCids[MAX\_NUM\_CCs][NUMBER\_OF\_UE\_MAX];

/// number of downlink active component carrier

int numactiveCCs[NUMBER\_OF\_UE\_MAX];

/// sorted uplink component carrier for the scheduler

int ordered\_ULCCids[MAX\_NUM\_CCs][NUMBER\_OF\_UE\_MAX];

/// number of uplink active component carrier

int numactiveULCCs[NUMBER\_OF\_UE\_MAX];

/// number of downlink active component carrier

uint8\_t dl\_CC\_bitmap[NUMBER\_OF\_UE\_MAX];

/// eNB to UE statistics

eNB\_UE\_STATS eNB\_UE\_stats[MAX\_NUM\_CCs][NUMBER\_OF\_UE\_MAX];

/// scheduling control info

UE\_sched\_ctrl UE\_sched\_ctrl[NUMBER\_OF\_UE\_MAX];

int next[NUMBER\_OF\_UE\_MAX];

int head;

int next\_ul[NUMBER\_OF\_UE\_MAX];

int head\_ul;

int avail;

int num\_UEs;

boolean\_t active[NUMBER\_OF\_UE\_MAX];

} UE\_list\_t;

## 4.10 mac\_rlc\_status\_resp\_t

\* \brief Primitive exchanged between RLC and MAC informing about the buffer occupancy of the RLC protocol instance.

\*/

typedef struct {

rlc\_buffer\_occupancy\_t bytes\_in\_buffer; /\*!< \brief Bytes buffered in RLC protocol instance. \*/

rlc\_buffer\_occupancy\_t pdus\_in\_buffer; /\*!< \brief Number of PDUs buffered in RLC protocol instance (OBSOLETE). \*/

frame\_t head\_sdu\_creation\_time; /\*!< \brief Head SDU creation time. \*/

sdu\_size\_t head\_sdu\_remaining\_size\_to\_send; /\*!< \brief remaining size of sdu: could be the total size or the remaining size of already segmented sdu \*/

boolean\_t head\_sdu\_is\_segmented; /\*!< \brief 0 if head SDU has not been segmented, 1 if already segmented \*/

} mac\_rlc\_status\_resp\_t;

# 5 Block

Table 4. lists the modules provided in \openairinterface5g\openair2\LAYER2\MAC

|  |  |  |
| --- | --- | --- |
| **Block** | **Description** | **3GPP Specification** |
| PHY\_MAC\_Interface | Primitives between MAC and PHY | N/A |
| DLSCH\_Process | Schedule info on DLSCH  …. | 5.4 UL-SCH data transfer |
| ULSCH\_Process | Schedule info on ULSCH  …. | 5.3 DL-SCH data trsansfer |
| RA\_Process | Schedule message related random access(Msg2、Msg4)  … | 5.1 Random Access procedure |
| SI\_Process | Schedule system information  … | 5.6 BCH reception |
| Configuration | Brief RRC Configuration primitive for PHY/MAC. Allows configuration of PHY/MAC resources based on System Information (SI), RRC Connection Setup and RRC Connection Reconfiguration messages.  ….. | 5.8 MAC reconfiguration  ….. |

Table x. MAC Modules and Functionality

## 5.1 PHY\_MAC\_Interface

### 5.1.1 Introduction

## 5.2 DLSCH\_Process

### 5.2.1 Introduction

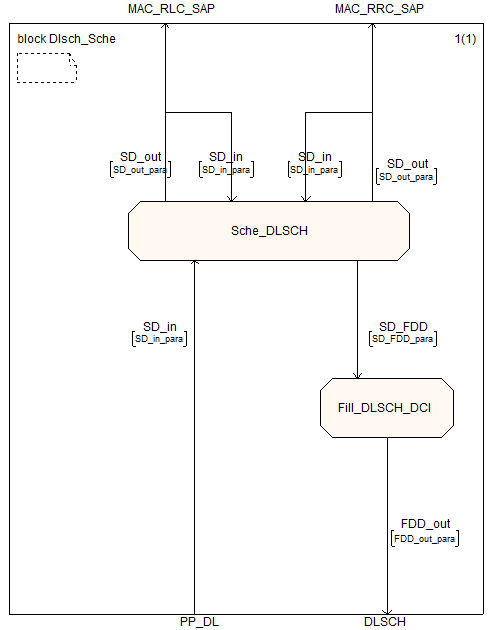


Fig.6 Block level of Dlsch\_Sche.

|  |  |  |  |
| --- | --- | --- | --- |
| **Process Name** | **Description** | **File Name** | **Function Name** |
| Sche\_DLSCH | Schedule info on DLSCH | eNB\_scheduler.c | schedule\_ue\_spec() |
| Fill\_DLSCH\_DCI | Second stage of DLSCH scheduling, after schedule\_SI, schedule\_RA and schedule\_dlsch have been called. This routine first allocates random frequency assignments for SI and RA SDUs using distributed VRB allocations and adds the corresponding DCI SDU to the DCI buffer for PHY. It then loops over the UE specific DCIs previously allocated and fills in the remaining DCI fields related to frequency allocation. It assumes localized allocation of type 0 (DCI.rah=0). The allocation is done for transmission modes 1, 2, 4. | eNB\_scheduler.c | fill\_DLSCH\_dci() |

Table 10. Processes in Dlsch\_Sche.

|  |  |  |
| --- | --- | --- |
| **Signal** | **Parameters** | |
| SD\_FDD | module\_idP, frame\_t frameP, subframeP, \* RBallocP, RA\_scheduledP, \* mbsfn\_flagP | |
| **Signal** | **Interface** | **Parameters** |
| SD\_in | PP\_DL | module\_idP, cooperation\_flag, frameP, subframeP(1, 2, 3, 4, 6, 7, 8, 9) |
| SD\_in | MAC\_RRC\_SAP | Get UE context status or RRC\_INACTIVE |
| SD\_out | MAC\_RRC\_SAP | module\_idP, rntiP |
| SD\_in | MAC\_RLC\_SAP | Get SDU length( bytes) from RLC through DCCH(SRB1)、DCCH1(SRB2)、DTCH |
| Get status response from RLC through DCCH(SRB1)、DCCH1(SRB2)、DTCH |
| SD\_out | MAC\_RLC\_SAP | module\_idP, rntiP, eNB\_index, frameP, enb\_flagP, MBMS\_flagP, channel\_idP, \*buffer\_pP |
| module\_idP, rntiP, eNB\_index, frameP, enb\_flagP, MBMS\_flagP, channel\_idP, tb\_sizeP |
| FDD\_out | DLSCH | DLSCH Message |

Table 11. Parameters of signal lines.

## 5.3 ULSCH\_Process

### 5.3.1 Introduction

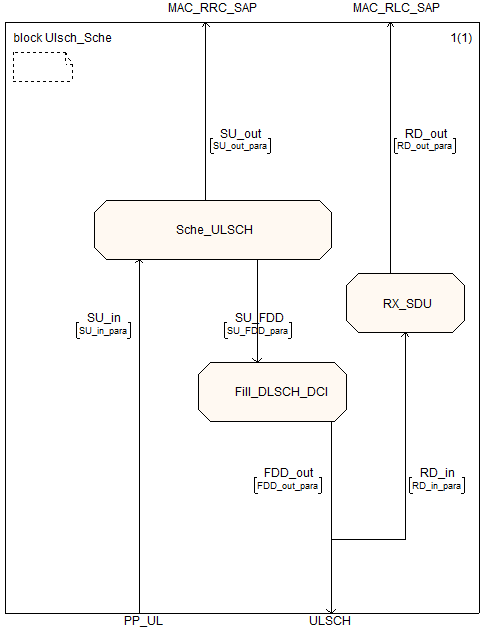


Fig.9 Block level of Ulsch\_sche.

|  |  |  |  |
| --- | --- | --- | --- |
| **Process Name** | **Description** | **File Name** | **Function Name** |
| Sche\_ULSCH | Schedule info on ULSCH | eNB\_scheduler.c | schedule\_ulsch() |
| Fill\_DLSCH\_DCI | Second stage of DLSCH scheduling, after schedule\_SI, schedule\_RA and schedule\_dlsch have been called. This routine first allocates random frequency assignments for SI and RA SDUs using distributed VRB allocations and adds the corresponding DCI SDU to the DCI buffer for PHY. It then loops over the UE specific DCIs previously allocated and fills in the remaining DCI fields related to frequency allocation. It assumes localized allocation of type 0 (DCI.rah=0). The allocation is done for transmission modes 1, 2, 4. | eNB\_scheduler.c | fill\_DLSCH\_dci() |
| RX\_SDU | Brief function to indicate a received SDU on ULSCH | eNB\_scheduler\_ulsch.c | rx\_sdu() |

Table 14. Processes in Ulsch\_Sche.

|  |  |  |
| --- | --- | --- |
| **Signal** | **Parameters** | |
| SD\_FDD | module\_idP, frame\_t frameP, subframeP, \* RBallocP, RA\_scheduledP, \* mbsfn\_flagP | |
| **Signal** | **Interface** | **Parameters** |
| SU\_in | PP\_UL | module\_idP, cooperation\_flag, frameP, subframeP(1, 2, 3, 4, 6, 8, 9), |
| SD\_out | MAC\_RRC\_SAP | Get UE context status or RRC\_INACTIVE |
| RD\_in | ULSCH | enb\_mod\_idP, CC\_idP, frameP, subframeP, rntiP, \*sduP, sdu\_lenP,  harq\_pidP, \*msg3\_flagP  Get status response from RLC through DCCH(SRB1)、DCCH1(SRB2)、DTCH |
| RD\_out | MAC\_RLC\_SAP | module\_idP, rntiP, eNB\_index, frameP, enb\_flagP, MBMS\_flagP, channel\_idP, \*buffer\_pP, tb\_sizeP, num\_tbP, \*crcs\_pP |
| FDD\_out | ULSCH | ULSCH Message |

Table 15. Parameters of signal lines.

## 5.4 RA\_Process

### 5.4.1 Introduction

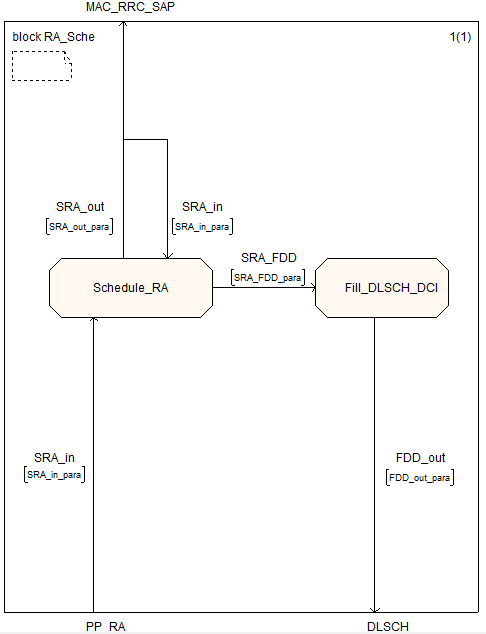


Fig.13 Block level of RA\_Sche.

|  |  |  |  |
| --- | --- | --- | --- |
| **Process Name** | **Description** | **File Name** | **Function Name** |
| Schedule\_RA | Schedule Msg2; Msg4 whenever it need. | eNB\_scheduler\_RA.c | Schedule\_RA() |
| Fill\_DLSCH\_dci | Add DCI format | eNB\_scheduler.c | fill\_DLSCH\_dci() |

Table.19 Processes in RA\_Sche.

|  |  |  |
| --- | --- | --- |
| **Signal** | **Parameters** | |
| SRA\_FDD | module\_idP, frameP, subframeP, \* RBallocP, RA\_scheduledP, \* mbsfn\_flagP | |
| **Signal** | **Interface** | **Parameters** |
| SRA\_out | MAC\_RRC\_SAP | module\_idP, CC\_idP, frameP,  srb\_idP, nb\_tbP, buffer\_pP,  eNB\_flagP, eNB\_indexP,  mbsfn\_sync\_areaP |
| SRA\_in | PP\_RA | module\_idP, cooperation\_flag, frameP, subframeP(0,5,6,7,9) |
| SRA\_in | MAC\_RRC\_SAP | Get SDU length from RRC through CCCH(SRB0) |
| FDD\_out | DLSCH | XXX |

Table 20. Parameters of signal lines.

## 5.5 SI\_Process

### 5.5.1 Introduction

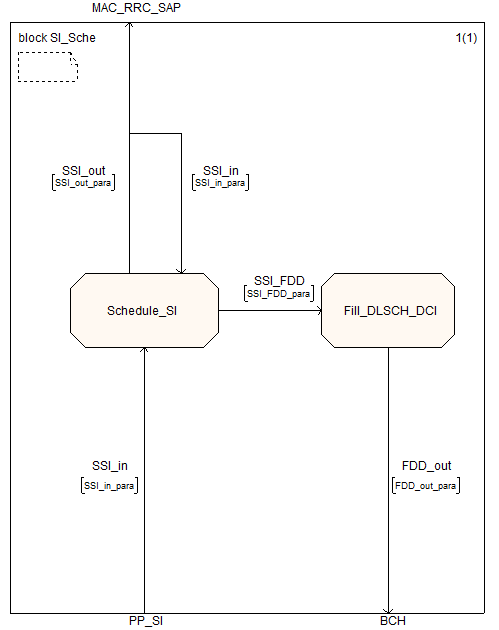


Fig.16 Block level of SI\_Sche.

|  |  |  |  |
| --- | --- | --- | --- |
| **Process Name** | **Description** | **File Name** | **Function Name** |
| Schedule\_SI | Schedule System Information | eNB\_scheduler\_bch.c | schedule\_SI() |
| Fill\_DLSCH\_dci | Add DCI format | eNB\_scheduler.c | fill\_DLSCH\_dci() |

Table 23. Processes in SI\_Sche.

|  |  |  |
| --- | --- | --- |
| **Signal** | **Parameters** | |
| SSI\_FDD | module\_idP, frameP, subframeP, \* RBallocP, RA\_scheduledP,\* mbsfn\_flagP | |
| **Signal** | **Interface** | **Parameters** |
| SSI\_in | PP\_SI | module\_idP, cooperation\_flag, frameP, subframeP(1, 2, 3, 4, 6, 7, 8, 9) |
| SSI\_in | MAC\_RRC\_SAP | Get SDU length from RRC through BCCH |
| SSI\_out | MAC\_RRC\_SAP | module\_idP, CC\_idP, frameP, srb\_idP, nb\_tbP, buffer\_pP,  eNB\_flagP, eNB\_indexP, mbsfn\_sync\_areaP |
| FDD\_out | BCH | BCH Message |

Table 24. Parameters of signal lines.

## 5.6 Configuration

### 5.6.1 Introduction

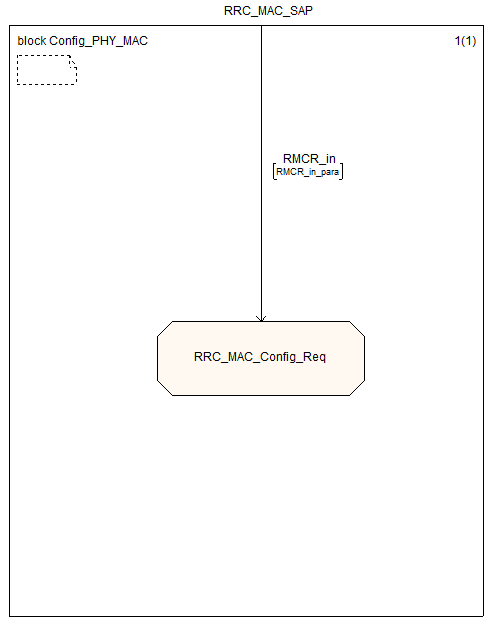


Fig.19 Block level of Config\_PHY\_MAC.

|  |  |  |  |
| --- | --- | --- | --- |
| **Process Name** | **Description** | **File Name** | **Function Name** |
| RRC\_MAC\_Config\_Req | Brief RRC Configuration primitive for PHY/MAC. Allows configuration of PHY/MAC resources based on System Information (SI), RRCConnectionSetup and RRCConnectionReconfiguration messages. | config.c | rrc\_mac\_config\_req() |

Table 27. Processes in Config\_PHY\_MAC.

|  |  |  |
| --- | --- | --- |
| **Signal** | **Parameters** | |
| N/A | N/A | |
| **Signal** | **Interface** | **Parameters** |
| RMCR\_in | RRC\_MAC\_SAP | module\_idP, CC\_id, eNB\_flag, rntiP, eNB\_index, \*radioResourceConfigCommon,  \*physicalConfigDedicated,  #ifdef Rel10  \*sCellToAddMod\_r10,  #endif  \*\*measObj, \*mac\_MainConfig, logicalChannelIdentity,  \*logicalChannelConfig, \*measGapConfig, \*tdd\_Config,  \*mobilityControlInfo, \*SIwindowsize, \*SIperiod,  \*ul\_CarrierFreq, \*ul\_Bandwidth,  \*additionalSpectrumEmission,  \*mbsfn\_SubframeConfigList  #ifdef Rel10  MBMS\_Flag,  \*mbsfn\_AreaInfoList,  \*pmch\_InfoList  #endif  #ifdef CBA  ,num\_active\_cba\_groups, cba\_rnti  #endif |

Table 28. Parameters of signal lines.