BMWLAB OAI Project Team

MAC-PHY Interface

Advisor : Cheng, Ray-Guang

Student : BMW LAB

Date : 2017/04

Outline

- **Downlink Control Information**
  - Different between LTE and NB-IoT
  - Detail Field Summary for DCI N0 N1 N2
  - NB-IoT DCI Structure example
- **FAPI-Style Interface for NB-IoT based on OAI**
  - 5 Step API
  - OAI old primitives in FAPI-Style interface
- **eNB MAC-PHY Interface in OAI**
  - Top level primitives
  - Detail definition to OAI primitive (by blocks)
    - Random Access
    - Config
    - Scheduling

Outline

- **Downlink Control Information**
  - Different between LTE and NB-IoT
  - Detail Field Summary for DCI N0 N1 N2
  - NB-IoT DCI Structure example
- FAPI-Style Interface for NB-IoT based on OAI
  - 5 Step API
  - OAI old primitives in FAPI-Style interface
- eNB MAC-PHY Interface in OAI
  - Top level primitives
  - Detail definition to OAI primitive (by blocks)
    - Random Access
    - Config
    - Scheduling

DCI in LTE vs NB-IoT

- In LTE, the DCI’s length should be determined by the bandwidth and the transmission mode (FDD or TDD), so the same format of the DCI may have the different length.
- There is no TDD mode in NB-IoT.
- DCI N1 may separate to 3 types of field.
- In NB-IoT, the length of DCI is fixed to
  - Format N0, N1 – 23bits
  - Format N2 – 15 bits

BMWLAB

DCI N0 – 23 bits

Field	Number of bits
Flag for format N0/format N1 differentiation	1
Subcarrier indication	6
Resource assignment	3
Scheduling delay	2
Modulation and coding scheme	4
Redundancy version	1
Repetition number	3
New data indicator	1
DCI subframe repetition number	2

BMWLAB

DCI N1 – 23 bits

For Initiated Random Access
CRC is scrambled with C-RNTI

Field	Number of bits
Flag for format N0/format N1 differentiation (set to 1)	1
NPDCCH order indicator (set to 1)	1
Starting number of NPRACH repetitions	2
Subcarrier indication of NPRACH	6
All the remaining bits (set to 1)	13

For Random Access MSG 2
CRC is scrambled with a RA-RNTI

Field	Number of bits
Flag for format N0/format N1 differentiation (set to 1)	1
NPDCCH order indicator (set to 0)	1
Scheduling delay	3
Resource assignment	3
Modulation and coding scheme	4
Repetition number	4
Reserved	5
DCI subframe repetition number	2

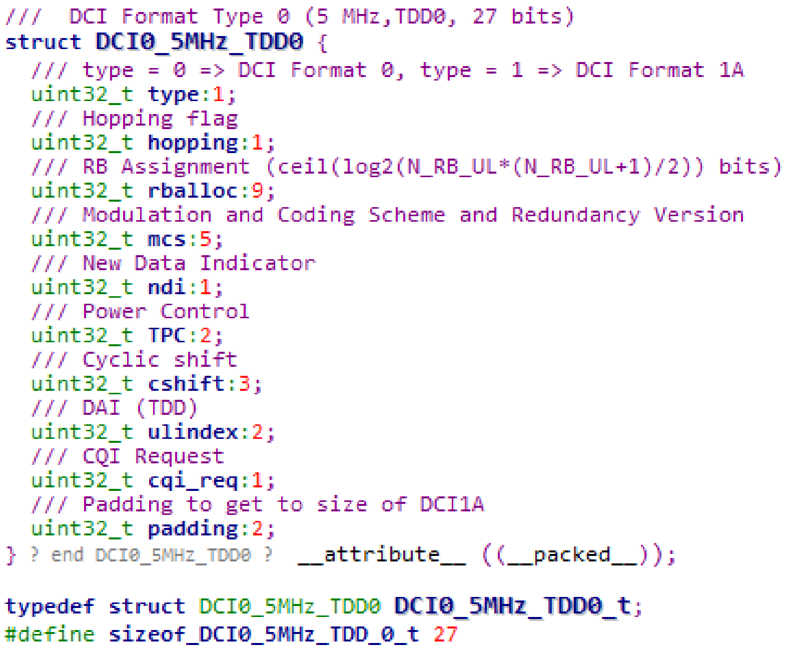
For Otherwise

Field	Number of bits
Flag for format N0/format N1 differentiation (set to 1)	1
NPDCCH order indicator (set to 0)	1
Scheduling delay	3
Resource assignment	3
Modulation and coding scheme	4
Repetition number	4
New data indicator	1
HARQ-ACK resource	4
DCI subframe repetition number	2

DCI N2 – 15 bits

Field	Number of bits
Flag for paging/direct indication differentiation (set to 1)	1
Resource assignment	3
Modulation and coding scheme	4
Repetition number	4
DCI subframe repetition number	3

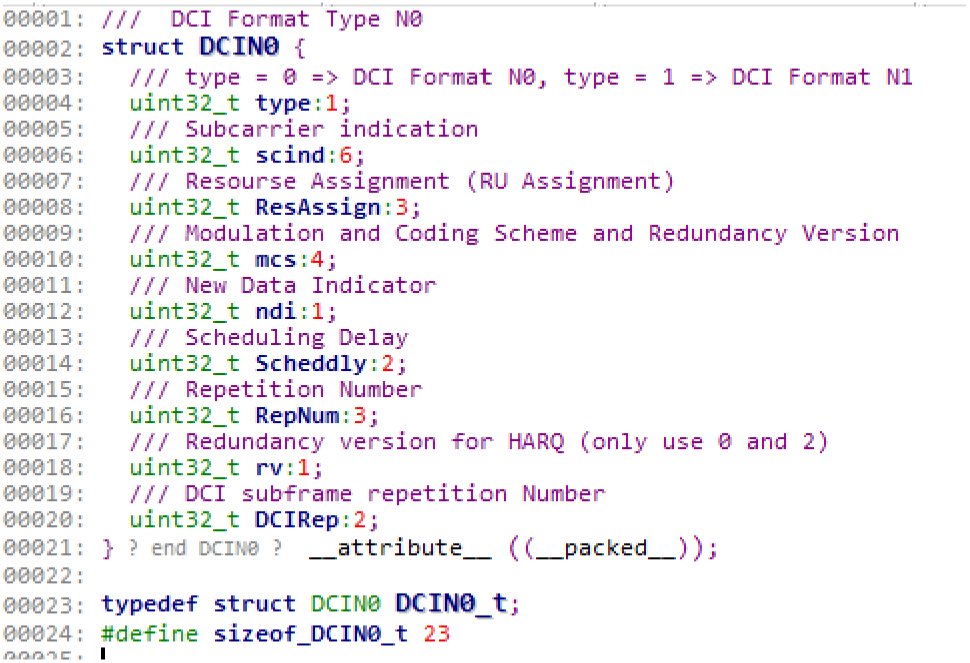
NB-IoT DCI structure example



```
///  DCI Format Type 0 (5 MHz,TDD0, 27 bits)
struct DCI0_5MHz_TDD0 {
  /// type = 0 => DCI Format 0, type = 1 => DCI Format 1A
  uint32_t type:1;
  /// Hopping flag
  uint32_t hopping:1;
  /// RB Assignment (ceil(log2(N_RB_UL*(N_RB_UL+1)/2)) bits)
  uint32_t rballoc:9;
  /// Modulation and Coding Scheme and Redundancy Version
  uint32_t mcs:5;
  /// New Data Indicator
  uint32_t ndi:1;
  /// Power Control
  uint32_t TPC:2;
  /// Cyclic shift
  uint32_t cshift:3;
  /// DAI (TDD)
  uint32_t ulindex:2;
  /// CQI Request
  uint32_t cqi_req:1;
  /// Padding to get to size of DCI1A
  uint32_t padding:2;
} ? end DCI0_5MHz_TDD0 ?  __attribute__ ((__packed__));

typedef struct DCI0_5MHz_TDD0 DCI0_5MHz_TDD0_t;
#define sizeof_DCI0_5MHz_TDD_0_t 27
```

OAI for LTE



```
///  DCI Format Type N0
struct DCIN0 {
  /// type = 0 => DCI Format N0, type = 1 => DCI Format N1
  uint32_t type:1;
  /// Subcarrier indication
  uint32_t scind:6;
  /// Resourse Assignment (RU Assignment)
  uint32_t ResAssign:3;
  /// Modulation and Coding Scheme and Redundancy Version
  uint32_t mcs:4;
  /// New Data Indicator
  uint32_t ndi:1;
  /// Scheduling Delay
  uint32_t Scheddly:2;
  /// Repetition Number
  uint32_t RepNum:3;
  /// Redundancy version for HARQ (only use 0 and 2)
  uint32_t rv:1;
  /// DCI subframe repetition Number
  uint32_t DCIRep:2;
} ? end DCIN0 ?  __attribute__ ((__packed__));

typedef struct DCIN0 DCIN0_t;
#define sizeof_DCIN0_t 23
```

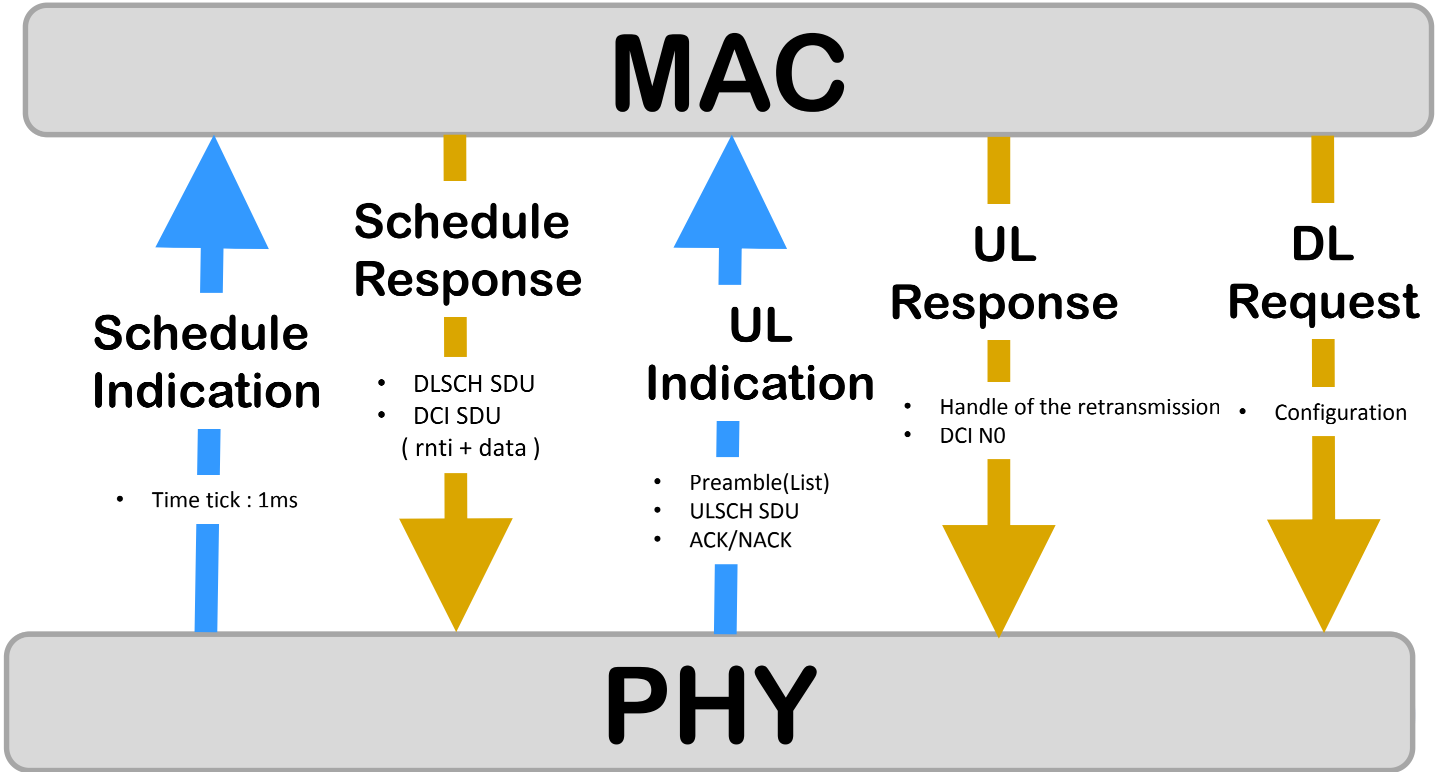
OAI for NB-IoT

Outline

- Downlink Control Information
  - Different between LTE and NB-IoT
  - Detail Field Summary for DCI N0 N1 N2
  - NB-IoT DCI Structure example
- **FAPI-Style Interface for NB-IoT based on OAI**
  - 5 Step API
  - OAI old primitives in FAPI-Style interface
- eNB MAC-PHY Interface in OAI
  - Top level primitives
  - Detail definition to OAI primitive (by blocks)
    - Random Access
    - Config
    - Scheduling

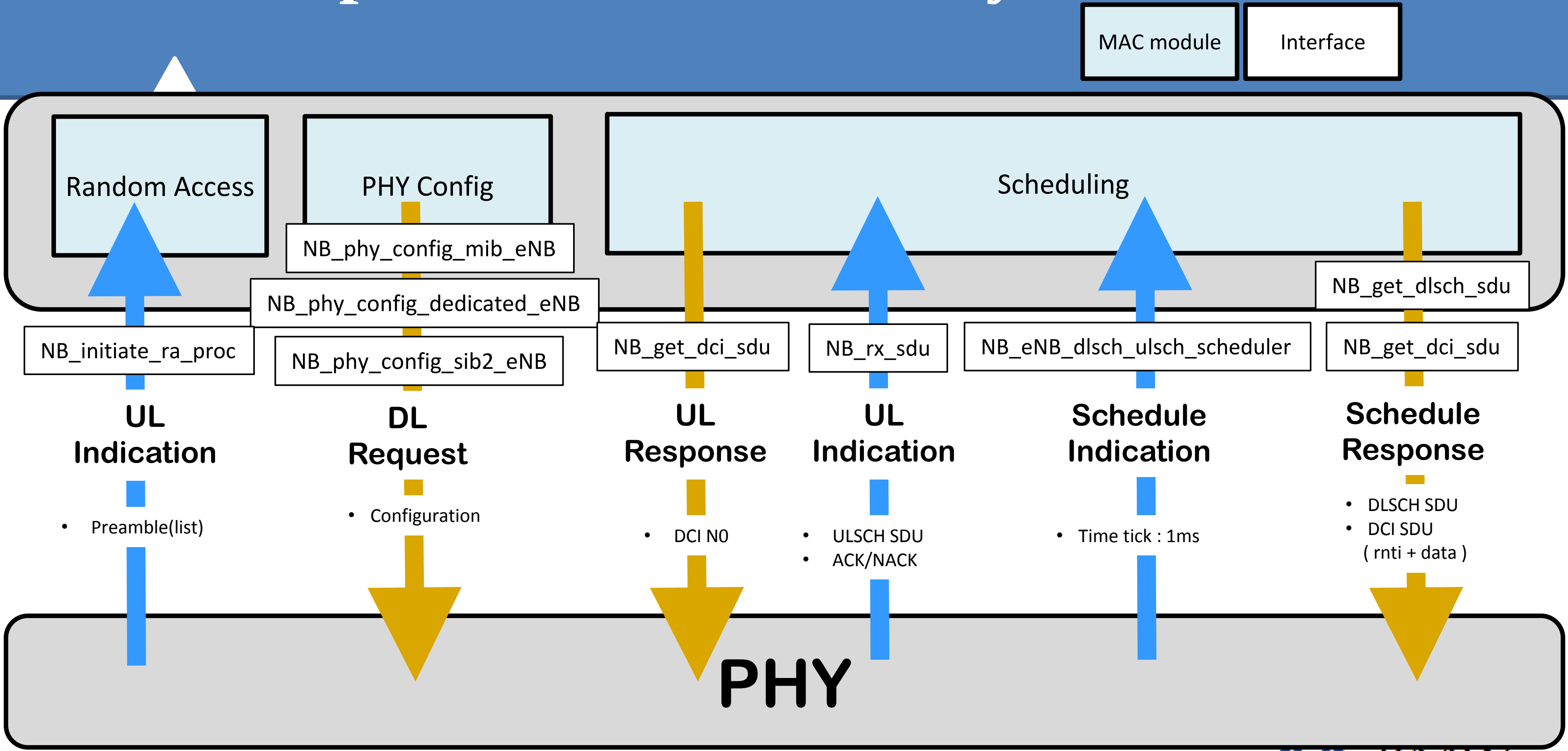
FAPI-Style Interface

- 5 steps API (general)



BMWLAB

OAI old primitives in FAPI-Style interface



MAC module

Interface

Random Access

PHY Config

Scheduling

NB_initiate_ra_proc

NB_phy_config_mib_eNB

NB_phy_config_dedicated_eNB

NB_phy_config_sib2_eNB

NB_get_dci_sdu

NB_rx_sdu

NB_eNB_dlsch_ulsch_scheduler

NB_get_dlsch_sdu

NB_get_dci_sdu

UL Indication

- Preamble(list)

DL Request

- Configuration

UL Response

- DCI N0

UL Indication

- ULSCH SDU
- ACK/NACK

Schedule Indication

- Time tick : 1ms

Schedule Response

- DLSCH SDU
- DCI SDU
(rnti + data)

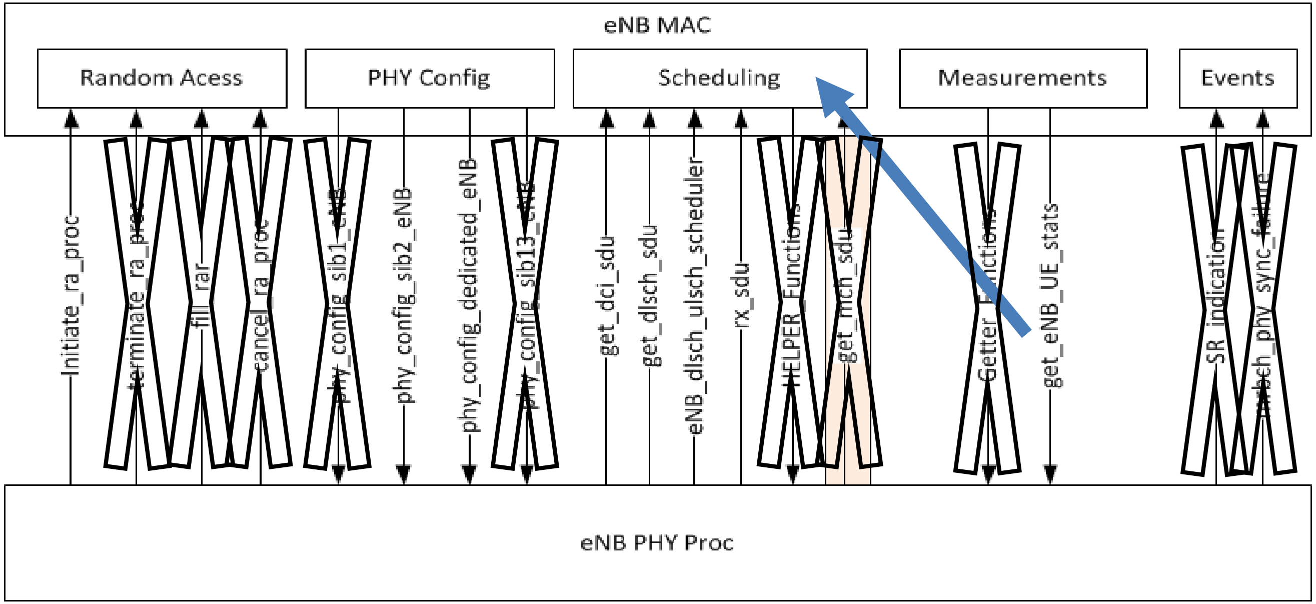
PHY

Outline

- Downlink Control Information
  - Different between LTE and NB-IoT
  - Detail Field Summary for DCI N0 N1 N2
  - NB-IoT DCI Structure example
- FAPI-Style Interface for NB-IoT based on OAI
  - 5 Step API
  - OAI old primitives in FAPI-Style interface
- **eNB MAC-PHY Interface in OAI**
  - Top level primitives
  - Detail definition to OAI primitive (by blocks)
    - Random Access
    - Config
    - Scheduling

eNB MAC-PHY interface

- Top level function (detail)
  - Openair2/PHY_INTERFACE/defs.h

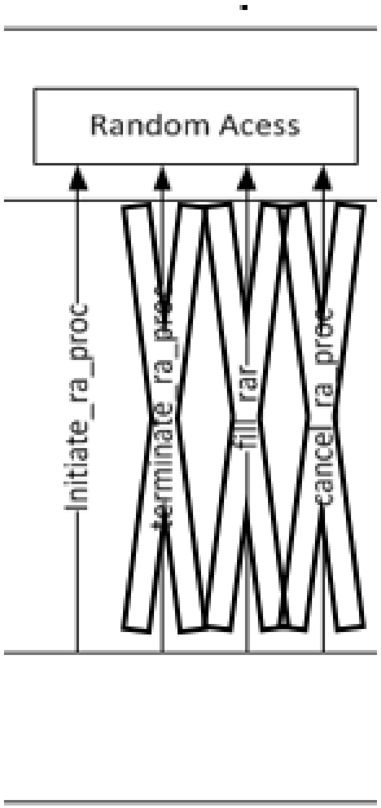


Note:

1. Add phy_config_MIB_eNB in PHY Config block
2. get_eNB_UE_stats should be in Scheduling block
3. Helper Functions should define in each side of the process (MAC or PHY)
4. SR_indication is no used for NB-IoT.
5. fill_rar will disappear on the new interface, this will be a normal DCI (with RA-RNTI) sent down by MAC.

BMWLAB

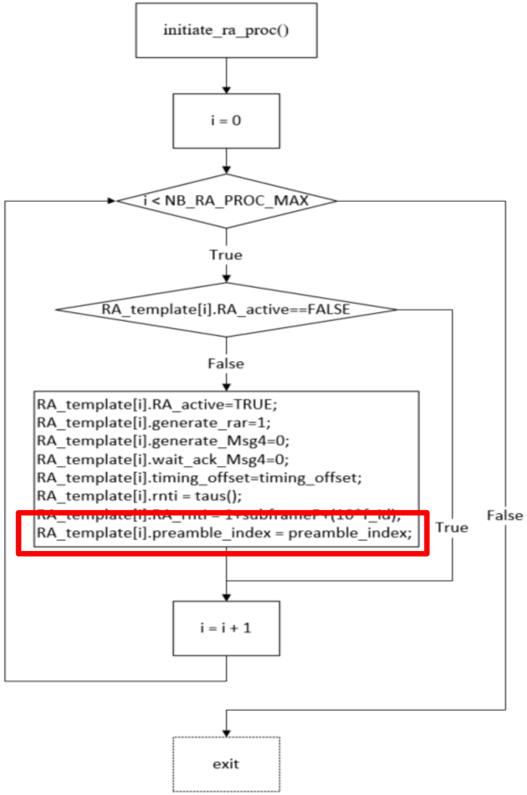
Random Access



- **[Modify]**Initiate_ra_proc:
  - Function to indicate a received preamble on PRACH. It initiates the RA procedure.
  - The preamble format has changed, and use the time and frequency domain to indicate the preamble.
- **[Modify]**fill_rar:
  - Function in eNB to fill RAR pdu when requested by PHY. This provides a single RAR SDU for the moment and returns the t-CRNTI.
  - The field of DCI for RAR and the RAR grant for MSG3 has changed.
- **[Re-use]**cancel_ra_proc:
  - Function to indicate a failed RA response. It removes all temporary variables related to the initial connection of a UE.

BMWLAB

initiate_ra_proc



initiate_ra_proc()

i = 0

i < NB_RA_PROC_MAX

True

RA_template[i].RA_active==FALSE

False

```
RA_template[i].RA_active=TRUE;
RA_template[i].generate_rar=1;
RA_template[i].generate_Msg4=0;
RA_template[i].wait_ack_Msg4=0;
RA_template[i].timing_offset=timing_offset;
RA_template[i].rnti = taus();
RA_template[i].RA_rnti = 1+subframe+(10*f_id);
RA_template[i].preamble_index = preamble_index;
```

True

False

i = i + 1

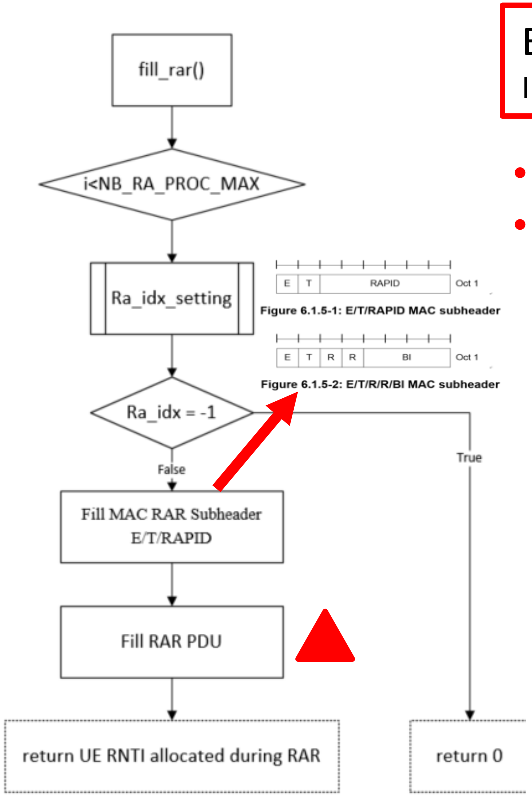
exit

Entry Point:

After rx_prach() in prach_procedures triggered by phy_procedures_eNB_common_RX , this primitive will start to initial a Random Access procedure.

- [Modify] RA_template_NB structure :
  - Preamble_freq domain[48]
  - Preamble_time domain
  - CE_level = 0 , 1 , 2
  - CE_level Mode A = 0,1 , Mode B = 2.

fill_rar


fill_rar()
i<NB_RA_PROC_MAX
Ra_idx_setting
E
T
RAPID
Oct 1
Figure 6.1.5-1: E/T/RAPID MAC subheader
E
T
R
R
BI
Oct 1
Figure 6.1.5-2: E/T/R/R/BI MAC subheader
Ra_idx = -1
False
True
Fill MAC RAR Subheader E/T/RAPID
Fill RAR PDU
return UE RNTI allocated during RAR
return 0

Entry Point

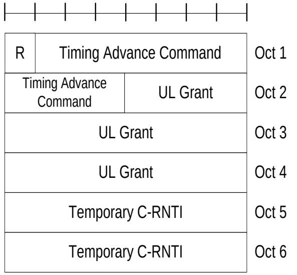
In pdsch_procedures, this function will run if ra_flag set to 1.

- [Modify] RAR PDU content
- [Modify] Fill RAR will disappear on the new interface, this will be a normal DCI (with RA-RNTI) sent down by MAC.

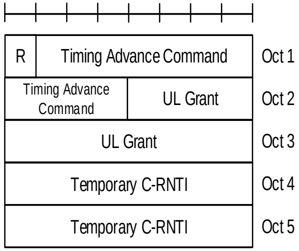
Table 6-2: Random Access Response Grant Content field size

DCI contents	CEmodeA	CEmodeB
Msg3 PUSCH narrowband index	N_{NB}^{index}	2
Msg3 PUSCH Resource allocation	4	3
Number of Repetitions for Msg3 PUSCH	2	3
MCS	3	0
TBS	0	2
TPC	3	0
CSI request	1	0
UL delay	1	0
Msg3/4 MPDCCH narrowband index	2	2
Zero padding	4 - N_{NB}^{index}	0
Total Nr-bits	20	12

CEmodeA

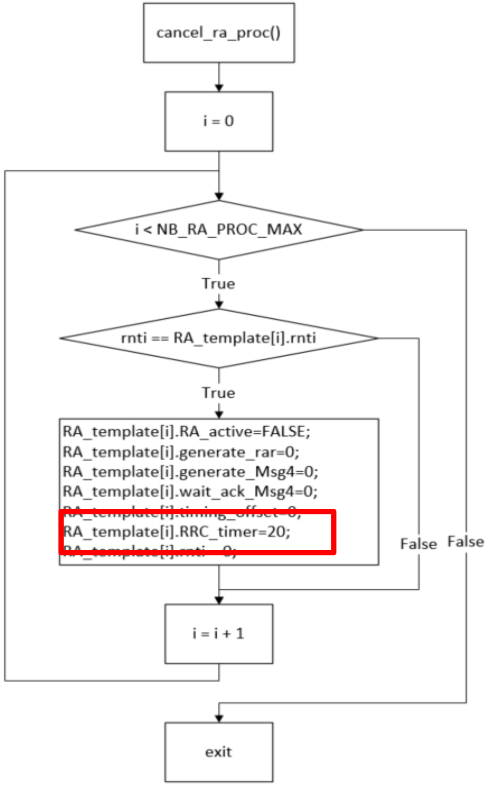


CEmodeB



BMWLAB

cancel_ra_proc


cancel_ra_proc()
i = 0
i < NB_RA_PROC_MAX
True
rnti == RA_template[i].rnti
True
RA_template[i].RA_active=FALSE;
RA_template[i].generate_rar=0;
RA_template[i].generate_Msg4=0;
RA_template[i].wait_ack_Msg4=0;
RA_template[i].timing_offset=0;
RA_template[i].RRC_timer=20;
RA_template[i].rnti=0;
False
False
i = i + 1
exit

Entry point

In phy_procedures_eNB_uespec_RX, this function will run if

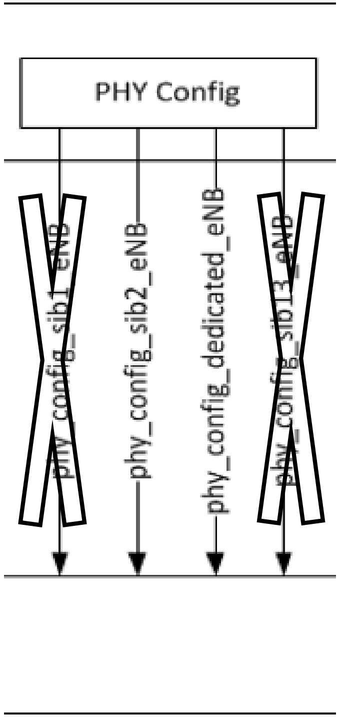
- maxHARQ_Msg3Tx reached, abandoning RA procedure for UE
- one-shot msg3 detection by MAC: empty PDU

In pdsch_procedures, this function will run if

- Max user count reached

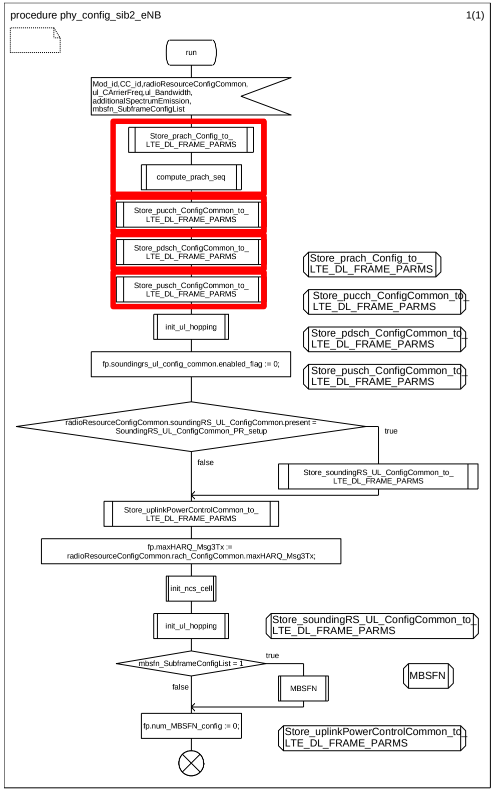
- [Delete] It’s no need for NB-IoT, MAC will definitely know that retransmission has been exhausted.

PHY Config



- [Delete]Phy_config_sib1_eNB:
  - SI window size & SI period
- [Modify]Phy_config_sib2_eNB:
  - Configuration of most frame parameters & channel
- [Modify]Phy_config_dedicated_eNB:
  - Configure PHY_VARS_eNB with components of physicalConfigDedicated
- [Add]Phy_config_mib_eNB:
  - Pass the important configuration from MIB for PHY.

phy_config_sibx_eNB


procedure phy_config_sib2_eNB
1(1)
run
Mod_id,CC_id,radioResourceConfigCommon,
ul_CArrierFreq,ul_Bandwidth,
additionalSpectrumEmission,
mbsfn_SubframeConfigList
Store_prach_Config_to_
LTE_DL_FRAME_PARMS
compute_prach_seq
Store_pucch_ConfigCommon_to_
LTE_DL_FRAME_PARMS
Store_pdsch_ConfigCommon_to_
LTE_DL_FRAME_PARMS
Store_pusch_ConfigCommon_to_
LTE_DL_FRAME_PARMS
init_ul_hopping
fp.soundingrs_ul_config_common.enabled_flag := 0;
Store_prach_Config_to_
LTE_DL_FRAME_PARMS
Store_pucch_ConfigCommon_to_
LTE_DL_FRAME_PARMS
Store_pdsch_ConfigCommon_to_
LTE_DL_FRAME_PARMS
Store_pusch_ConfigCommon_to_
LTE_DL_FRAME_PARMS
radioResourceConfigCommon.soundingRS_UL_ConfigCommon.present =
SoundingRS_UL_ConfigCommon_PR_setup
true
false
Store_soundingRS_UL_ConfigCommon_to_
LTE_DL_FRAME_PARMS
Store_uplinkPowerControlCommon_to_
LTE_DL_FRAME_PARMS
fp.maxHARQ_Msg3Tx :=
radioResourceConfigCommon.rach_ConfigCommon.maxHARQ_Msg3Tx;
init_ncs_cell
init_ul_hopping
Store_soundingRS_UL_ConfigCommon_to_
LTE_DL_FRAME_PARMS
mbsfn_SubframeConfigList = 1
true
false
MBSFN
MBSFN
fp.num_MBSFN_config := 0;
Store_uplinkPowerControlCommon_to_
LTE_DL_FRAME_PARMS

Entry Point

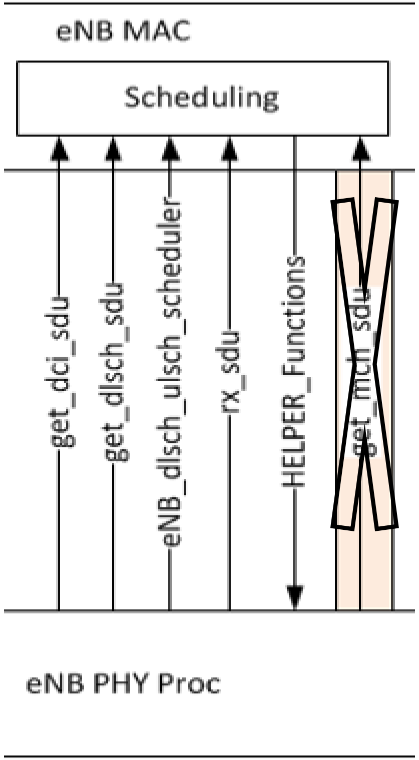
In phy_config_sibx_eNB, functions will called by

- rrc_mac_config_req call in the initial statement.

- phy_config_sib1_eNB
  - Use for TDD mode primitive (not use in NB-IoT)
- phy_config_sib2_eNB
  - [Modify] PRACH config -> NPRACH config
  - [Modify] PDSCH config -> NPDSCH config
  - [Modify] PUSCH config -> NPUSCH config
  - Modify the terms which are NB-PHY implementation needed according to the Physical layer design.
  - Reference to TS36.331-d30 Chapter 6.7.3.1 【SystemInformationBlockType2-NB】
- phy_config_sib13_eNB
  - Multicast mode not be used in NB-IoT

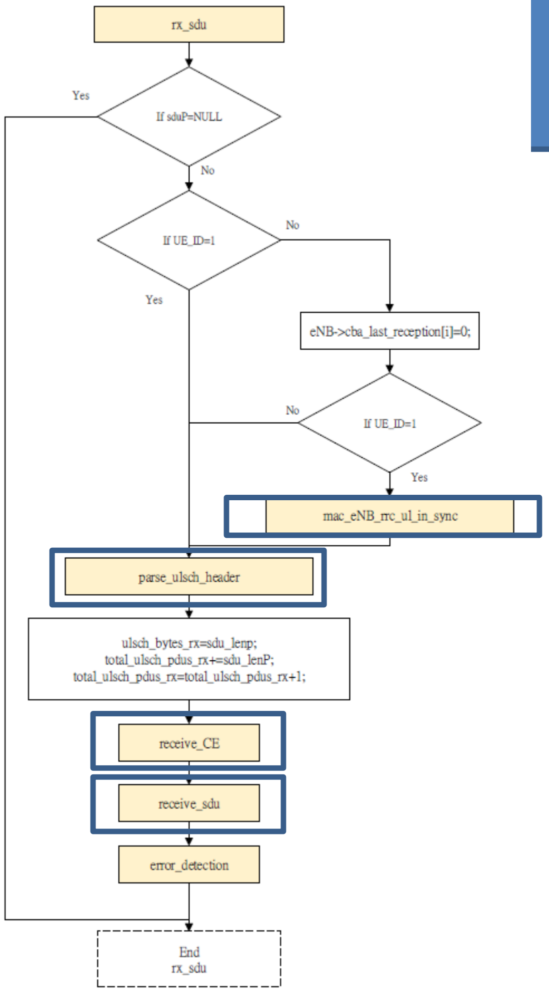
BMWLAB

Scheduling



- **[Re-use]**get_dci_sdu:
  - retrieve result of scheduling (DCI) in current subframe. Can be called an arbitrary number of times after eNB_dlsch_ulsch_scheduler.
- **[Re-use]**get_dlsch_sdu:
  - PHY get downlink data from MAC layer.
- **[Re-use]**eNB_dlsch_ulsch_scheduler:
  - Function to trigger the eNB scheduling procedure.
- **[Modify]NB_rx_sdu**:
  - MAC get Uplink data from PHY layer.

rx_sdu



Entry Point:

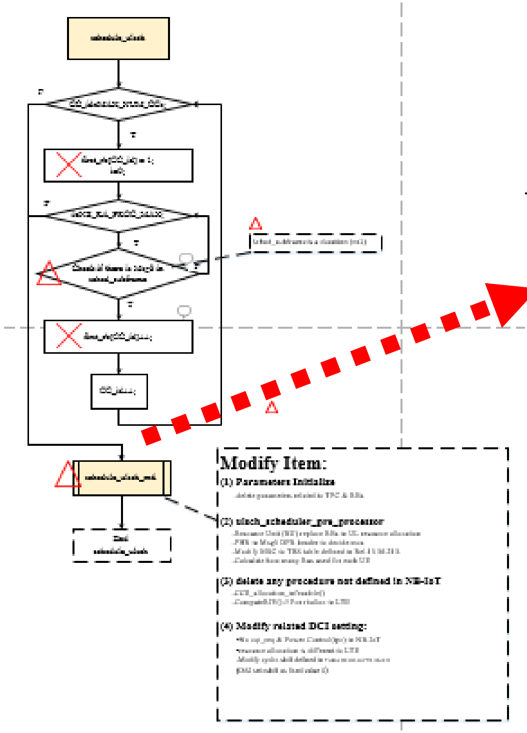
When rxtx thread created, this function will run if this subframe is set for uplink scheduled. After passing some limitation (check if is uplink scheduled, if turbo code can be decoded correctly…), this function will start to transport data from PHY to upper layer.

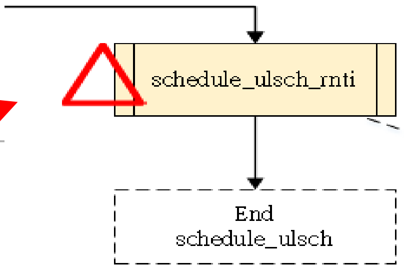
- [Re-use]mac_eNB_rrc_ul_in_sync:
  - Get UE context for RLF (check if out of sync) from RRC layer.
- [Re-use]parse_ulsch_header:
  - Parse uplink share channel header from MAC PDU.
- [Modify]receive_CE:
  - This part will change the correspond information determined by the case of Control Element(BSR,CRNTI…).
  - In NB-IoT, it will check DPR (DV and PHR) when Msg3 receiving.
- [Modify]receive_sdu:
  - This part include receiving sdu by the case of logical channel to use different primitives to transport data to RLC or RRC.
    - Include mac_rlc_data_ind() mac_rrc_data_ind()
  - In NB-IoT, there is no DTCH channel for the case.

schedule_ulsch

Entry Point:

When rxtx thread created, this function will run in subframe n for each subframe and it’s set for uplink scheduled. This function will start to schedule UL data for schduled subframe(n+4).


schedule_ulsch
Check if there is Msg3 in sched_subframe
schedule_ulsch_rnti
End
schedule_ulsch
Modify Item:
(1) Parameters Initialize
(2) ulsch_scheduler_pre_processor
(3) delete any procedure not defined in NB-IoT
(4) Modify related DCI setting:



Modify Item:

(1) Parameters Initialize

-delete parameters related to TPC & RBs

(2) ulsch_scheduler_pre_processor

-Resource Unit (RU) replace RBs in UL resource allocation
-PHR in Msg3 DPR header to decide mcs
-Modify MSC to TBS table defined in Rel.13 36.213.
-Calculate how many Rus used for each UE

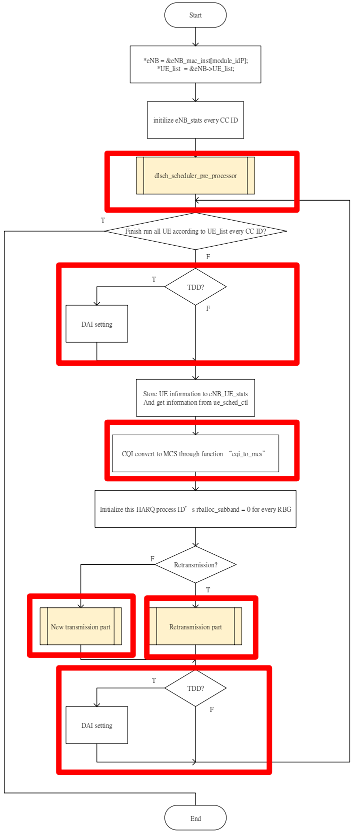
(3) delete any procedure not defined in NB-IoT

-CCE_allocation_infeasible()
-ComputeRIV() // For rballoc in LTE

(4) Modify related DCI setting:

-No cqi_req & Power Control(tpc) in NB-IoT
-resource allocation is different to LTE
-Modify cyclic shift defined in Table 10.1.4.1.2-3 TS 36.211
(OAI set cshift as fixed value 0)

schedule_ue_spec


Start
*eNB = &eNB_mac_inst[module_idP];
*UE_list = &eNB->UE_list;
initilize eNB_stats every CC ID
dlsch_scheduler_pre_processor
Finish run all UE according to UE_list every CC ID?
T
F
TDD?
T
F
DAI setting
Store UE information to eNB_UE_stats
And get information from ue_sched_ctl
CQI convert to MCS through function "cqi_to_mcs"
Initialize this HARQ process ID' s rballoc_subband = 0 for every RBG
Retransmission?
F
T
New transmission part
Retransmission part
TDD?
T
F
DAI setting
End

Entry point:

In LTE, trigger downlink scheduler per subframe.
In NB-IoT, we only need to trigger downlink scheduler one time per PP (PDCCH period).

- [Modify] : dlsch_scheduler_pre_processor
  - Need a new way to implement NB-IoT resource allocation.
- [Delete] TDD relative:
  - NB-IoT do not support TDD option.
- [Delete]CQI relative:
  - No CQI implementation in NB-IoT
- [Modify]Retransmission part:
  - This part need a new way to implement NB-IoT resource allocation
- [Modify]New transmission part:
  - This part need a new way to implement NB-IoT resource allocation
  - DTCH part is not support in NB-IoT C-Plane solution.
- [Modify] DCI
  - DCI have different format in NB-IoT. Add new DCI format N0, N1 and N2.