The MAC layer is responsible for
controlling how data is transmitted and
received over the air interface between
the gNB and User Equipment (UEs).

Start

Calling external files

void
*nrmac_stats_thread
(void *arg)

void
*nrmac_stats_thread
(void *arg)

size_t dump_mac_stats
(gNB_MAC_INST
*gNB, char *output, size_t
strlen, bool reset_rsrp)

static void
mac_rrc_init(gNB_MAC_INST
*mac, ngran_node_t
node_type)

mac_top_init_gNB(ngran_node_t node_type,
NR_ServingCellConfigCommon_t *scc,
NR_ServingCellConfig_t *scd,
const nr_mac_config_t *config)

nr_mac_send_f1_setup_req

void
*nrmac_stats_thread
(void *arg)

open file
nrMAC_stats.log

file exists ──False──

True

oai_exit = 0?

True

Intialize p to output

Lock 'sched_lock'

Intialize p to output

Call
'dump_mac_stats'
and update p

Unlock 'sched_lock'

Add newline to output

Call 'print_meas_log'
and update p

Write output to file

Flush file buffer and
sleep for 1 second

Reset file pointer to
the beginning

```
                            void                          size_t dump_mac_stats
                      *nrmac_stats_thread                    (gNB_MAC_INST
                         (void *arg)                     *gNB, char *output, size_t
                                                          strlen, bool reset_rsrp)


                         Get UE (user
                        equipment) list                    Initialize pointers
                        gNB -> UE_info.list

  Print "Cannot open
  rMAC_stats.log, error"                                   Ensure scheduler
                          Iterate over UE                      locked


                                                                                      "mutual exclusi
                          Clear mac_stats                   Lock UE info Mutex          control acc
                            (memset)                                                 programming en
                                                                                     can access the re
   close file                                                                                  ar
                                                            Iterate over UEs

                             return
                                                          Retrieve scheduler
    return                                                  Ctrl and Stats


                                                             Calculate RSRP


                                                            Print UE RNTI and
                                                              CU-UE-ID
                      nr_mac_send_f1_setup_req

                                                                                False
                                                              F1 UE Data                    Print "(none)"
                                                              exists?
                         Retrieve MAC
                         instance from
                         RC.nrmac[0]                          True

                                                            Print Sec.
                        Assert MAC instance                  UE ID
                        DevAssert(mac)

                                                                                False
                                                            Sync status                  Print out-of-syn
                        Assert MAC instance                 in-sync?                        status
                        DevAssert(mac)

                                                             True

                            Return                          Print sync
                                                            status


                                                            Print CSI
                                                            report


                                                            Print SRS
                                                            Stats
```

```
static void
mac_rrc_init(gNB_MAC_INST
    *mac, ngran_node_t
    node_type)
```
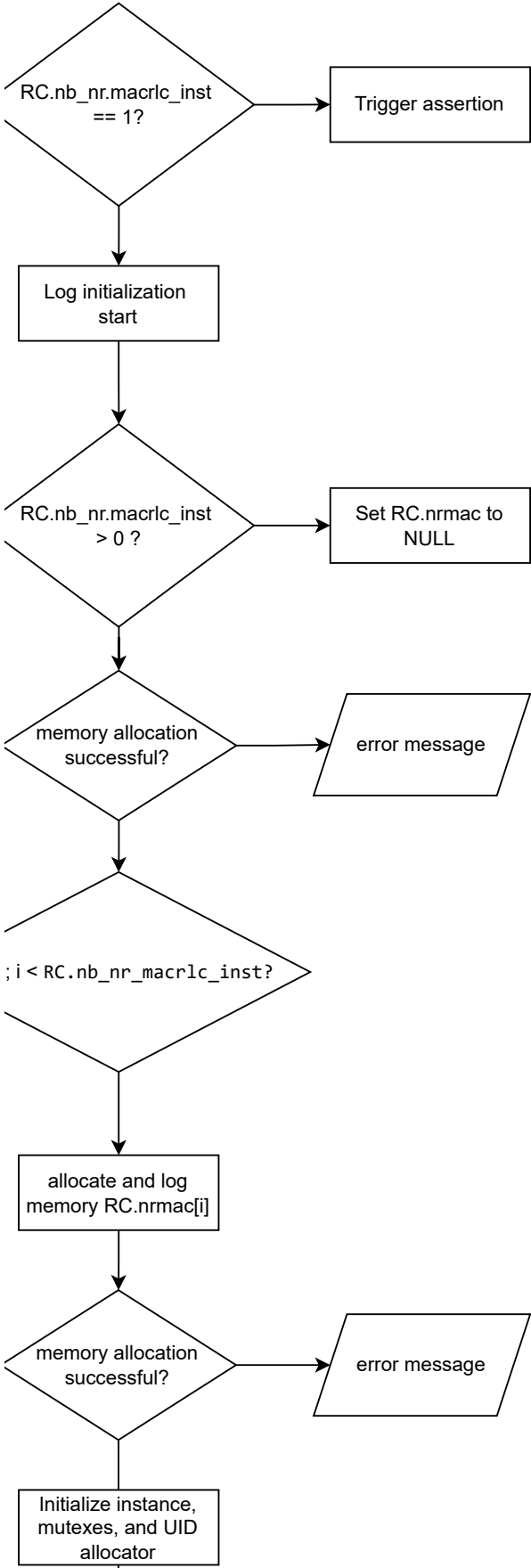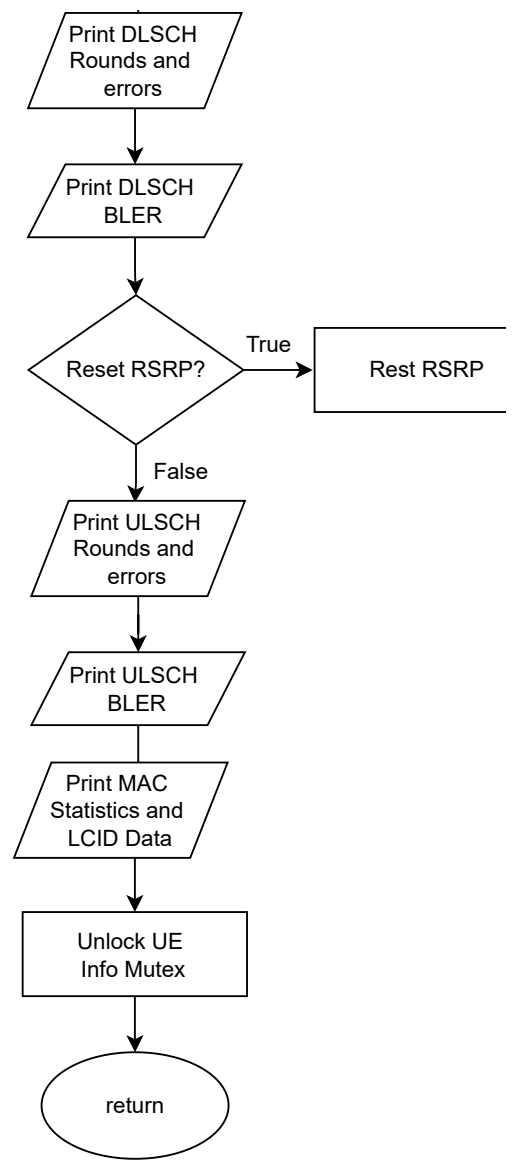
```
mac_top_in
    NR_Serv
    NR_S
    const
```

Is node type ngran_gNB_CU? — True → Trigger assertion failure

False

Is node type ngran_gNB_DU? — True → Initialize MAC RRC for gNB DU mac_rrc_ul_flap_init

False

Is node type ngran_gNB — True → Initialize MAC RRC for gNB DU mac_rrc_ul_flap_init

False

Trigger assertion failure : unknown node type

return
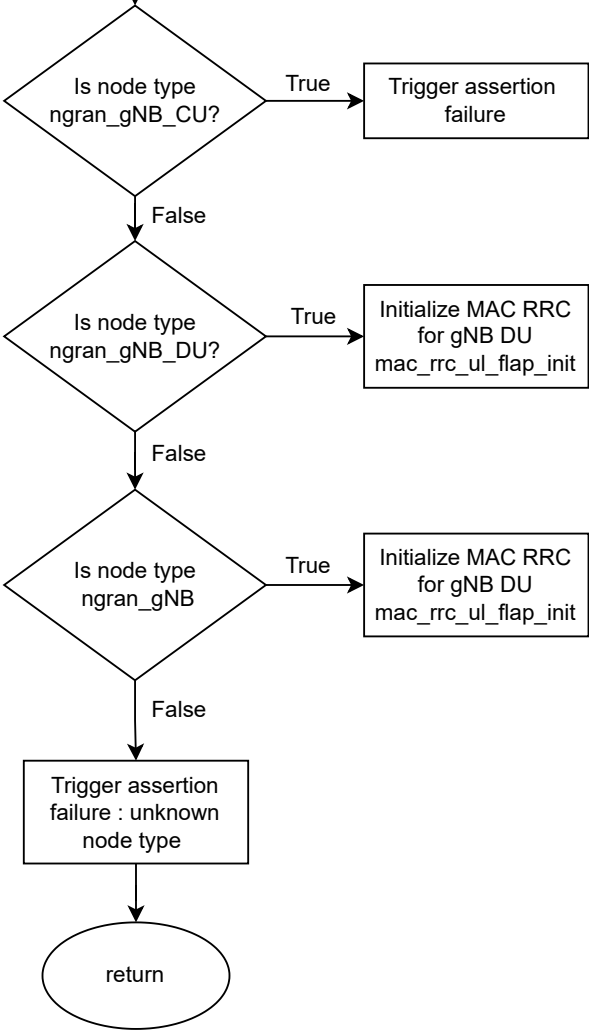
ion," is a synchronization primitive used to
ess to a shared resource in concurrent
nvironments. It ensures that only one thread
esource at a time, preventing race conditions
nd ensuring data consistency.

ic

i = 0
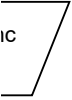
```
it_gNB(ngran_node_t node_type,
ingCellConfigCommon_t *scc,
ervingCellConfig_t *scd,
: nr_mac_config_t *config)
```

RC.nb_nr.macrlc_inst == 1? → Trigger assertion

Log initialization start

RC.nb_nr.macrlc_inst > 0 ? → Set RC.nrmac to NULL

memory allocation successful? → error message

; i < RC.nb_nr_macrlc_inst?

allocate and log memory RC.nrmac[i]

memory allocation successful? → error message

Initialize instance, mutexes, and UID allocator

```
┌─────────────────┐
 / Print DLSCH    /
/  Rounds and    /
/  errors       /
└─────────────────┘
        │
        ▼
┌─────────────────┐
 / Print DLSCH    /
/  BLER          /
└─────────────────┘
        │
        ▼
      ◇ Reset RSRP? ◇ ──True──► ┌──────────────┐
        │                        │  Rest RSRP   │
      False                      └──────────────┘
        │
        ▼
┌─────────────────┐
 / Print ULSCH    /
/  Rounds and    /
/  errors        /
└─────────────────┘
        │
        ▼
┌─────────────────┐
 / Print ULSCH    /
/  BLER          /
└─────────────────┘
        │
        ▼
┌─────────────────┐
 / Print MAC      /
/  Statistics and /
/  LCID Data     /
└─────────────────┘
        │
        ▼
┌─────────────────┐
│  Unlock UE      │
│  Info Mutex     │
└─────────────────┘
        │
        ▼
     (  return  )
```

- **RLC Module Initialization -->** Ensures reliable data transmission by handling segmentation, reassembly, error correction, and flow contro. As for, **PDCP Layer Initialization -->** Provides header compression, encryption, integrity protection, and in-sequence delivery of packets. It is initialized conditionally based on specific parameters.

```
┌─────────────────────┐
│ Set Pre-processors  │
│ and start thread    │
└─────────────────────┘
           │
           ▼
┌─────────────────────┐
│ initialize MAC RRC  │
└─────────────────────┘
           │
           ▼
┌─────────────────────┐
│ Initialize RLC      │
│ module and PDCP     │
│ layer               │
└─────────────────────┘
           │
           ▼
┌─────────────────────┐
│ Configure noS1      │
│ Bearer              │
└─────────────────────┘
           │
           ▼
┌─────────────────────┐
│ initialize active UEs│
│ and F1 UE Data      │
└─────────────────────┘
           │
           ▼
┌─────────────────────┐
│ Set Random Seed     │
└─────────────────────┘
           │
           ▼
┌─────────────────────┐
│ Trigger PHY         │
│ Initialization      │
└─────────────────────┘
           │
           ▼
       ╭─────────╮
       │ Return  │
       ╰─────────╯
```