Core testing infrastructure - Feature #4510
Support EGPRS RLC/MAC blocks in ttcn3

04/23/2020 08:01 PM - pespin

Status: In Progress
Priority: Normal
Assignee: pespin
Category:
Target version:
Start date: 04/23/2020
Due date:
% Done: 40%

Description
Most of the required information can be found in 3GPP TS 44.060:
https://www.etsi.org/deliver/etsi_ts/144000_144099/144060/15.00.00_60/ts_144060v150000p.pdf

First of all, EGPRS Data Ul/Dl block structures and encode/decode functions are missing in library/RLCMAC_Types.ttcn. Check "10.3a EGPRS RLC data blocks and RLC/MAC headers" for that.

Different MCS values have different header structure in those blocks, so we probably need to have different encode/decode functions based on data length, and then have a union to be able to pass them together?

It seems there's also a MCS-0 Control Dl block header, which at least I think it's not needed for now but we need to check what's that used for.

We also need to implement the missing CSN1 Control messages related to EGPRS, using L/H features which afaiu are not available since we use titan 6.6.1.

Related issues:
Related to OsmoPCU - Feature #4488: have uplink/downlink tests for all CS/MCS New 04/07/2020
Related to OsmoPCU - Feature #4286: Support configuration using 8PSK on dowlink... New 11/28/2019
Related to OsmoPCU - Bug #4338: Add EGPRS tests to TTCN3 PCU_Tests_RAW In Progress 12/23/2019

History
#1 - 04/23/2020 08:02 PM - pespin
- Related to Feature #4488: have uplink/downlink tests for all CS/MCS added

#2 - 04/23/2020 08:03 PM - pespin
- Related to Feature #4286: Support configuration using 8PSK on dowlink while staying GMSK-only on uplink added

#3 - 04/23/2020 08:03 PM - pespin
- Related to Bug #4338: Add EGPRS tests to TTCN3 PCU_Tests_RAW added

#4 - 04/24/2020 07:24 AM - laforge
On Thu, Apr 23, 2020 at 08:02:29PM +0000, pespin [REDMINE] wrote:

We also need to implement the missing CSN1 Control messages related to EGPRS, using L/H features which afaiu are not available since we use titan 6.6.1.

6.6.1 is the latest release, and according to the git history it contains both
231012acb95c68cba7e029eb6901ac76216dcd5b and 0a4bb8b632bdfba76767b778fe7b1a758462181
i.e. the L/H annotation for the RAW codec support.

#5 - 04/24/2020 11:22 AM - pespin

Sorry, it was late yesterday when I wrote it, I meant to write "using L/H features which afaiu are available since we use titan 6.6.1."

#6 - 04/24/2020 01:42 PM - pespin
- Description updated
I started adding EGPRS DL Data block decoding to our C++ decoder in library/RLCMAC_EncDec.cc which is already handling GPRS ones. WIP in pespin/pcu-cap-egprs. So from TTCN3 point of view it looks like:

```c
/* TS 44.060 10.3a.1.1 EGPRS downlink RLC data block, manual c++ encoder/decoder */
type record EgprsDlMacDataHeader {
    uint2_t header_type, /* Set internally by decoder */
    + uint5_t tfi,
    + MacRrbp rrbp,
    + BIT2 esp,
    + uint3_t usf,
    + uint14_t bsn1,
    + uint8_t bsn2_offset,
    + uint2_t pr, /* power reduction */
    + uint2_t spb,
    + uint4_t cps,
    + BIT8 spare1,
    + BIT8 spare2,
    + BIT8 spare3,
    + BIT1 fbi,
    + boolean e
} with {
    variant (e) "FIELDLENGTH(1)"
};
/* Manual C++ Decoder: */
type record RlcmacDlEgprsDataBlock {
    EgprsDlMacDataHeader mac_hdr,
    EgprsLlcBlocks blocks
} with {
    variant ""
};
/* TS 44.060 10.2.2 */
type record UlMacDataHeader {
/* Octet 0 */
module RLCMAC_Types {
union RlcmacDlBlock {
    RlcmacDlDataBlock data,
    RlcmacDlEgprsDataBlock data_egprs,
    RlcmacDlCtrlBlock ctrl
} with {
    variant *TAG(data, mac_hdr.mac_hdr.payload_type = MAC_PT_RLC_DATA;
    ctrl, mac_hdr.payload_type = MAC_PT_RLCMAC_NO_OPT;
    ctrl, mac_hdr.payload_type = MAC_PT_RLCMAC_OPT)
    + ctrl, {mac_hdr.payload_type = MAC_PT_RLCMAC_NO_OPT,
        mac_hdr.payload_type = MAC_PT_RLCMAC_OPT};
    + data_egprs, {mac_hdr.header_type = 1,
        mac_hdr.header_type = 2,
        mac_hdr.header_type = 3}
    + mac_hdr.header_type = 3}
};
```

I'll probably need to add a MCS value to data_egprs_mac_hdr.mcs to be able to known how to encode it into proper mcs (and its header type).

Some stuff it's still not working on the decoding side but TTCn3 test is already tagging it as data_egprs when using the generic RlcMacDlData decode function:

```c
{ data_egprs := { mac_hdr := { header_type := 3, tfi := 0, rrbp := RRPB_Nplus13_mod_2715648 (0), esp := '1'0'B, usf := 7, bsn1 := 0, bsn2_offset := 0, pr := 0, spb := 0, cps := 1l, spare1 := '0'B, spare2 := <unbound>, spare3 := <unbound>, fbi := '1'B, e := false }, blocks := { { hdr := { length_ind := 3, more := true, e := true }, payload := '000016'O } } } }
```

Initial support to decode EGPRS data block with header type 3 (MCS1,2,3,4) merged in osmo-ttcn3-hacks:

commit 372af7a116a961fcf7008a9f8e00ad0e81a809ae
Author: Pau Espin Pedrol <pespin@sysmocom.de>
Date: Mon Apr 27 17:32:01 2020 +0200
library/RLCMAC: Add partial support for EGPRS data block encoding/decoding

* RlcmacULBlock and RlcmacDLBlock gain a new union field "egprs_data", which is chosen when msg contains an egprs data block. Hence one can use same structure for both gprs/egprs data and simply check "ischosen(block.data_egprs)" to know whether it contains egprs or gprs data block.
* C++ code in RLCMAC_EncDec.cc takes care of encoding and decoding of each data header type and exposes a generic ttcn3 struct "ULMacDataHeader" and "DLMacDataHeader". Decoded header type can be found in mac_hdr.header_type. This can be used together with CPS to get the MCS of the message received. Similarly, the encoder will use the same field to know how to encode the ttcn3 structure.
* In RLCMAC_EncDec.cc order of functions has been ordered to split between encoding and decoding, and inside these split between UL and DL messages.
* Only encoding of UL HeaderType3 and decoding of DL HeaderType3 is implemented so far in RLCMAC_EncDec.cc. However, all code is already arranged and functions prepared (with FIXME fprintf) to easily add the missing header types once needed.
* Actually only the decoding of DL HeaderType3 has been tested to work so far. Encoding may still be missing to octet-align the data block after the header. All these will be fixed once a test using them exists.

Change-Id: I2bc4f877a5e17c57ffa8cf05565dc8593b45aaa8

#9 - 04/29/2020 02:19 PM - pespin
- Status changed from New to In Progress
- % Done changed from 0 to 40

#10 - 04/30/2020 06:27 PM - pespin

I added a bunch more fixes, improvements and new enc/dec support in RLCMAC_EncDec.cpp:

remote: https://gerrit.osmocom.org/c/osmo-ttcn3-hacks/+17985 cosmetic: RLCMAC_EncDec.cc: fix typos in comments
remote: https://gerrit.osmocom.org/c/osmo-ttcn3-hacks/+17986 RLCMAC_EncDec.cc: dec_RlcmacUL(Egprs)DataBlock : fix tli and pfi uninitialize...
remote: https://gerrit.osmocom.org/c/osmo-ttcn3-hacks/+17987 RLCMAC_EncDec.cc: Fix wrong descriptors passed to RAW encoder
remote: https://gerrit.osmocom.org/c/osmo-ttcn3-hacks/+17988 RLCMAC_EncDec.cc: Fix encoding of tfi and rsb fields in Egprs UL HeaderType3
remote: https://gerrit.osmocom.org/c/osmo-ttcn3-hacks/+17989 RLCMAC_EncDec.cc: Use copied structure as other parts of the function
remote: https://gerrit.osmocom.org/c/osmo-ttcn3-hacks/+17990 RLCMAC_EncDec.cc: Support decoding Egprs UL Data HeaderType2
remote: https://gerrit.osmocom.org/c/osmo-ttcn3-hacks/+17991 RLCMAC_Templates: Add functions to convert HeaderType<->MCS<->CPS

I'm still keeping some stuff in my branch pespin/egprs-encdec a selftest to encode+decode UL data blocks (and send it to PCU so they can be cross-checked with Wireshark dissector).

As of current master, encoding is not working correctly because data block is not "unaligned" against the rlcmac block. I have some improvements regarding that issue in my branch, but I'm still facing issues with the resulting data block not being the same after de-coding again (both in wireshark and ttcn3 egprs decoder show same payload, but it's not the same that was initially encoded).