OsmoBTS - Bug #1854

11-bit RACH support breaks default 8-bit RACH: collisions are possible

11/18/2016 10:02 PM - laforge

Status:	Resolved	**Start date:**	11/18/2016
Priority:	Urgent	**Due date:**	
Assignee:	fixeria	**% Done:**	100%
Category:	osmo-bts-trx		
Target version:	osmo-bts-trx refresh		
Spec Reference:	3GPP TS 05.02, section 5.2.7		

Description

osmo-bts-trx currently can only do 8bit RACH, unlike other BTS models

Related issues:

Related to OsmoPCU - Bug #1548: 11bit RACH support	**Resolved**	**02/23/2016**
Related to OsmoTRX - Feature #3054: Extended (11-bit) RACH support in OsmoTRX	**Stalled**	**03/10/2018**

Associated revisions

Revision 322cfc49 - 02/07/2018 07:45 PM - max

Add helper to get BCC from BSIC

Change-Id: Ib539a8739d53ab60d8fadffcef38152b82a28498
Related: OS#1854

Revision de9e2092 - 02/14/2019 03:48 PM - max

osmo-bts-trx: add extended (11-bit) RACH support

Attempt to decode incoming RACH burst as 11-bit first
and fallback to 8-bit if unsuccessful.

Change-Id: Ia28741603636406744e5e22ffff1fb7a9689955a
Related: OS#1854

Revision a4785902 - 03/05/2019 10:30 AM - max

coding: check gsm0503_rach_*() results

Check return value of RACH encode/decode functions and fail test on
unexpected results.

Change-Id: I41bfa808e3c064a11152e7ce8ee77a01d38a0744
Related: OS#1854

Revision 5f9e42a8 - 05/09/2019 04:22 PM - Vadim Yanitskiy

osmo-bts-trx: distinguish 11-bit Access Bursts by synch. sequence

Thanks to both TC_rach_content and TC_rach_count TTCN-3 test cases,
it was discovered that there are possible collisions when trying
to decode a regular 8-bit Access Burst as an 11-bit one. This is
exactly what we are doing in rx_rach_fn():

```
- calling gsm0503_rach_ext_decode_ber() first,
  - if it failed, falling-back to gsm0503_rach_decode_ber().
```

With default BSIC=63, the following 8-bit RA values are being
misinterpreted as 11-bit Access Bursts:

```
Successfully decoded 8-bit (0x00) RACH as 11-bit (0x0000): bsic=0x3f
  Successfully decoded 8-bit (0xbe) RACH as 11-bit (0x0388): bsic=0x3f
  Successfully decoded 8-bit (0xcf) RACH as 11-bit (0x0036): bsic=0x3f
```

According to 3GPP TS 05.02, section 5.2.7, there are two alternative
synch. (training) sequences for Access Bursts: TS1 & TS2. By default,
TS0 synch. sequence is used, unless explicitly stated otherwise

(see 3GPP TS 04.60).

According to 3GPP TS 04.60, section 11.2.5a, the EGPRS capability
can be indicated by the MS using one of the alternative training
sequences (i.e. TS1 or TS2) and the 11-bit RACH coding scheme.

In other words, knowing the synch. sequence of a received Access
Burst would allow to decide whether it's extended (11-bit)
or a regular (8-bit) one. As a result, we would avoid possible
collisions and save some CPU power.

Unfortunately, due to the limitations of the current TRXD protocol,
there is no easy way to expose such information from the transceiver.
A proper solution would be to extend the TRX protocol, but for now,
let's do the synch. sequence detection in rx_rach_fn(). As soon as
the TRX protocol is extended with info about the synch. sequence,
this code would serve for the backwards-compatibility.

This change makes the both TC_rach_content and TC_rach_count happy,
as well as the new TC_pcu_ext_rach_content() test case aimed to
verify extended (11-bit) Access Burst decoding.

Related (TTCN-3) I8fe156aeac9de3dc1e71a4950821d4942ba9a253
Change-Id: Ibb6d27c6589965c8b59a6d2598a7c43fd860f284
Related: OS#1854

Revision dc48fd88 - 07/16/2019 04:16 AM - Vadim Yanitskiy

osmo-bts-trx/scheduler: rx_rach_fn(): use optional TSC info from TRX

TSC (Training Sequence Code) is an optional parameter of the UL burst
indication. We need this information in order to decide whether an
Access Burst is 11-bit encoded or not (see OS#1854).

If this information is absent, we try to correlate the received synch.
sequence with the known ones (3GPP TS 05.02, section 5.2.7), and
fall-back to the default TS0 if it fails.

Since the new TRXD header version, the training sequence code is
indicated by the transceiver. Let's use it!

Change-Id: I1e654a2e49cb83c5f1e6249c0de688f99bc466b0
Related: OS#1854, OS#4006

History

#1 - 12/03/2017 10:23 AM - laforge

- Assignee set to msuraev

#2 - 12/04/2017 11:02 AM - msuraev

- Related to Bug #1548: 11bit RACH support added

#3 - 12/04/2017 12:04 PM - msuraev

- Status changed from New to In Progress

Encoding/decoding support for 11-bit RACH for libosmocoding is available in gerrit 5062.

#4 - 01/17/2018 04:34 PM - msuraev

- % Done changed from 0 to 10

Support was merged to libosmocoding. README update is in gerrit 5833. The next step is to implement it in osmo-bts-trx and test.

#5 - 01/30/2018 09:17 AM - msuraev

- Status changed from In Progress to Stalled

#6 - 02/21/2018 11:34 PM - msuraev

The tests with gerrit 6315 is not working with current master of omopcu. We should re-test with sysmobts andimplement missing pieces.

#7 - 03/01/2018 11:14 PM - laforge

- Assignee changed from msuraev to sysmocom

#8 - 03/10/2018 07:22 AM - laforge

- Related to Feature #3054: Extended (11-bit) RACH support in OsmoTRX added

#9 - 03/10/2018 07:23 AM - laforge

Please note not even OsmoTRX currently has support for this, see #3054

#10 - 10/17/2018 09:44 AM - laforge

- Assignee changed from sysmocom to msuraev

#11 - 11/29/2018 12:41 PM - laforge

see also https://gerrit.osmocom.org/#/c/osmo-bts/+/5833/

#12 - 02/21/2019 04:39 PM - msuraev

- Status changed from Stalled to In Progress

- % Done changed from 10 to 40

After merging https://gerrit.osmocom.org/c/osmo-bts/+/6315 and corresponding https://gerrit.osmocom.org/c/osmo-trx/+/11423 and related OsmoTRX patches there's TTCN-3 failure in TC_rach_content() and TC_rach_count() due to some (about 16 out of 1000) RACH being "lost". So far I'm unable to reproduce this using real phone. I'm not sure yet what's so special about the lost RACH (e. g. '0xCF') which makes a difference or whether rach content itself has anything to do with it at all. Investigation is ongoing.

The error constantly appears only with '0xCF', '0xBE' and '0x00' RA values for different FN.

#13 - 03/26/2019 03:29 PM - msuraev

- Status changed from In Progress to Stalled

- % Done changed from 40 to 50

Related https://gerrit.osmocom.org/c/osmo-ttcn3-hacks/+/13128 might be useful for local testing.
Once the issue is fixed, https://gerrit.osmocom.org/c/osmo-bts/+/5833 should be merged as well.

#14 - 04/20/2019 11:04 PM - fixeria

- Tracker changed from Feature to Bug

- Status changed from Stalled to In Progress

- Assignee changed from msuraev to fixeria

- % Done changed from 50 to 80

- Spec Reference set to 3GPP TS 05.02, section 5.2.7

> The error constantly appears only with '0xCF', '0xBE' and '0x00' RA values for different FN.

Please see: https://gerrit.osmocom.org/#/c/osmo-bts/+/13723/

Changing to 'Bug' since merging this feature (without proper testing) introduced a regression.

#15 - 04/21/2019 08:33 PM - fixeria

- Subject changed from 11bit RACH support in osmo-bts-trx to 11-bit RACH support breaks default 8-bit RACH: collisions are possible

- Priority changed from Normal to Urgent

> Please see: https://gerrit.osmocom.org/#/c/osmo-bts/+/13723/

I've got an idea: what if I overcomplicated the task with the synch. sequence detection? What if changing the order of both gsm0503_rach_ext_decode_ber() and gsm0503_rach_decode_ber() would be enough? I did a quick test, and yes! Changing the order vice versa makes both TC_rach_content() and TC_rach_count() TTCN-3 test cases happy. However...

To be 100% sure, I decided to write a simple collision test. The key idea is to encode all possible 8-bit RA values for all possible BSIC values, and attempt to decode them as 11-bit RACH. This replicates the current behaviour. Additionally, this test encodes all possible 11-bit RA values for all possible BSIC values, and attempts to decode them as 8-bit RACH. Please see an attached file.

```
=== Looking for possible collisions: decoding 8-bit RACH as 11-bit
Successfully decoded 8-bit (0xc0) RACH as 11-bit (0x0500): bsic=0x00
Successfully decoded 8-bit (0x06) RACH as 11-bit (0x0184): bsic=0x01
Successfully decoded 8-bit (0x7e) RACH as 11-bit (0x04cf): bsic=0x01
Successfully decoded 8-bit (0x80) RACH as 11-bit (0x0700): bsic=0x01
Successfully decoded 8-bit (0x82) RACH as 11-bit (0x04fc): bsic=0x01
Successfully decoded 8-bit (0xa5) RACH as 11-bit (0x02f3): bsic=0x01
Successfully decoded 8-bit (0xfa) RACH as 11-bit (0x0737): bsic=0x01
Successfully decoded 8-bit (0x3a) RACH as 11-bit (0x0237): bsic=0x02
Successfully decoded 8-bit (0x6a) RACH as 11-bit (0x001c): bsic=0x02
Successfully decoded 8-bit (0x77) RACH as 11-bit (0x057d): bsic=0x02
Successfully decoded 8-bit (0xf6) RACH as 11-bit (0x0764): bsic=0x02
Successfully decoded 8-bit (0x0a) RACH as 11-bit (0x030c): bsic=0x03
Successfully decoded 8-bit (0x19) RACH as 11-bit (0x0435): bsic=0x03
Successfully decoded 8-bit (0x3a) RACH as 11-bit (0x0237): bsic=0x03
Successfully decoded 8-bit (0x8e) RACH as 11-bit (0x011e): bsic=0x03
Successfully decoded 8-bit (0xa5) RACH as 11-bit (0x02f3): bsic=0x03
Successfully decoded 8-bit (0x0f) RACH as 11-bit (0x03ed): bsic=0x04
Successfully decoded 8-bit (0x37) RACH as 11-bit (0x027d): bsic=0x04
Successfully decoded 8-bit (0x42) RACH as 11-bit (0x0317): bsic=0x04
...

=== Looking for possible collisions: decoding 11-bit RACH as 8-bit
Successfully decoded 11-bit (0x0067) RACH as 8-bit (0x86): bsic=0x00
Successfully decoded 11-bit (0x0100) RACH as 8-bit (0xc0): bsic=0x00
Successfully decoded 11-bit (0x0146) RACH as 8-bit (0x87): bsic=0x00
Successfully decoded 11-bit (0x0197) RACH as 8-bit (0xda): bsic=0x00
Successfully decoded 11-bit (0x0354) RACH as 8-bit (0x03): bsic=0x00
Successfully decoded 11-bit (0x03a5) RACH as 8-bit (0x2b): bsic=0x00
Successfully decoded 11-bit (0x061a) RACH as 8-bit (0xbe): bsic=0x00
Successfully decoded 11-bit (0x0628) RACH as 8-bit (0xe6): bsic=0x00
Successfully decoded 11-bit (0x062a) RACH as 8-bit (0x4c): bsic=0x00
Successfully decoded 11-bit (0x0657) RACH as 8-bit (0x6a): bsic=0x00
Successfully decoded 11-bit (0x068c) RACH as 8-bit (0x2a): bsic=0x00
Successfully decoded 11-bit (0x06c0) RACH as 8-bit (0xe0): bsic=0x00
Successfully decoded 11-bit (0x0714) RACH as 8-bit (0xcc): bsic=0x00
Successfully decoded 11-bit (0x0734) RACH as 8-bit (0x4c): bsic=0x00
Successfully decoded 11-bit (0x0748) RACH as 8-bit (0xde): bsic=0x00
Successfully decoded 11-bit (0x078a) RACH as 8-bit (0x5d): bsic=0x00
Successfully decoded 11-bit (0x07bc) RACH as 8-bit (0xc2): bsic=0x00
Successfully decoded 11-bit (0x07f0) RACH as 8-bit (0x08): bsic=0x00
Successfully decoded 11-bit (0x07f1) RACH as 8-bit (0x5d): bsic=0x00
Successfully decoded 11-bit (0x0006) RACH as 8-bit (0x3e): bsic=0x01
Successfully decoded 11-bit (0x0164) RACH as 8-bit (0x36): bsic=0x01
Successfully decoded 11-bit (0x0173) RACH as 8-bit (0xe5): bsic=0x01
Successfully decoded 11-bit (0x0184) RACH as 8-bit (0x06): bsic=0x01
Successfully decoded 11-bit (0x0185) RACH as 8-bit (0x53): bsic=0x01
Successfully decoded 11-bit (0x0212) RACH as 8-bit (0xc0): bsic=0x01
Successfully decoded 11-bit (0x026f) RACH as 8-bit (0x33): bsic=0x01
Successfully decoded 11-bit (0x0276) RACH as 8-bit (0x1f): bsic=0x01
...
```

The results show that collisions are possible in both cases, regardless of the ordering. Changing the order would make the situation even worse:

```
=== Results:
 - decoding 8-bit RACH as 11-bit: 260
 - decoding 11-bit RACH as 8-bit: 1961
```

This analysis should have been done before merging extended (11-bit) RACH support.

I think the proposed solution (i.e. distinguishing by the synch. sequence: https://gerrit.osmocom.org/#/c/osmo-bts/+/13723/) is probably the only valid way, and we need to merge it as soon as possible. With default BSIC=63, the following 8-bit RA values are misinterpreted as 11-bit RACH bursts:

```
Successfully decoded 8-bit (0x00) RACH as 11-bit (0x0000): bsic=0x3f
Successfully decoded 8-bit (0xbe) RACH as 11-bit (0x0388): bsic=0x3f
Successfully decoded 8-bit (0xcf) RACH as 11-bit (0x0036): bsic=0x3f
```

Recently I introduced the extended (11-bit) RACH support in OsmocomBB/trxcon:

https://gerrit.osmocom.org/#/c/osmocom-bb/+/13731/
https://gerrit.osmocom.org/#/c/osmocom-bb/+/13732/

so the missing part is a couple of TTCN-3 test cases.

The last (but quite important) question for me: may 11-bit RACH bursts use the default TS0 synch. sequence, as regular 8-bit RACH does? And may a regular 8-bit RACH use the additional TS1 / TS2 synch. sequences? I couldn't find the answer in specifications, but if yes, then I don't really know, how else can we distinguish 8-bit and 11-bit RACH?

#16 - 04/21/2019 08:51 PM - fixeria

- File rach_test.c added

Adding the forgotten attachment.

#17 - 04/21/2019 10:09 PM - fixeria

> The last (but quite important) question for me [...]

Ok, I've finally found the answer in 3GPP TS 04.60, section 11.2.5a. The EGPRS capability can be indicated by the MS using the alternative training sequences (i.e. TS1 or TS2) and the 11-bit RACH coding scheme. Therefore, the if we detect TS0 - it's a generic 8-bit Access Burst, otherwise it's an extended 11-bit Access Burst. The proposed solution is correct.

#18 - 04/22/2019 12:06 AM - fixeria

- Status changed from In Progress to Feedback

- % Done changed from 80 to 100

> so the missing part is a couple of TTCN-3 test cases.

Please see:

https://gerrit.osmocom.org/#/c/osmo-ttcn3-hacks/+/13734/
https://gerrit.osmocom.org/#/c/osmo-ttcn3-hacks/+/13735/

Everything seems to work just fine. The last thing I need to do is to write a proper commit message for:

https://gerrit.osmocom.org/#/c/osmo-bts/+/13723/

so we can merge it then and close this issue.

#19 - 05/09/2019 06:44 PM - fixeria

- Status changed from Feedback to Resolved

The fix for osmo-bts-trx has been merged.

Files

rach_test.c	2.79 KB	04/21/2019	fixeria