TSG-RAN Working Group 1 meeting No. 7
Aug. 30-Sept. 3, Hannover, Germany

3GPP/TSGR1#7(99)d10

Source: Motorola

Title: Text proposal on RACH message scrambling

This text proposal gives updated text for Sections 4.3.3.1 and 4.3.3.5 for TS25.213 V2.1.2. The changes clarify the definition of the RACH scrambling code numbers and the generation of the RACH message scrambling codes.

4.3.3.1 Preamble scrambling code

The scrambling code for the preamble part is as follows.

The code generating method is the same as for the real part of the long codes on dedicated channels. Only the first 4096 chips of the code are used for preamble spreading with the chip rate of 3.84 Mchip/s. The long code c1 for the in-phase component is used directly on both in phase and quadrature branches without offset between branches. The preamble scrambling code is defined as the position wise modulo 2 sum of 4096 chips segments of two binary m-sequences generated by means of two generator polynomials of degree 25. Let x and y be the two m-sequences respectively. The x sequence is constructed using the primitive (over GF(2)) polynomial $X^{25}+X^3+1$. The y sequence is constructed using the polynomial $X^{25}+X^3+X^2+X+1$. The resulting sequences thus constitute segments of a set of Gold sequences.

Let ~~n_7~~ n_{23}… n_0 be the binary representation of the code number *n* (decimal) with n_0 being the least significant bit. Code numbers between 0 and 255 are used for the random access channel. The *m*-sequences x_n and *y* are constructed as:

Initial conditions:

~~$x_n(0)=n_0$, $x_n(1)= n_1$, … $=x_n(6)= n_6$, $x_n(7)= n_7$, $x_n(8)= 0$,…, $x_n(22)= 0$,$x_n(23)= 1$, $x_n(24)=0$~~

$x_n(0)=n_0$, $x_n(1)= n_1$, … $=x_n(22)= n_{22}$, $x_n(23)= n_{23}$, $x_n(24)=1$

$y(0)=y(1)= \ldots =y(23)= y(24)=1$

Recursive definition of subsequent symbols:

$x_n(i+25) =x_n(i+3) + x_n(i)$ *modulo 2, i=0,…,* 4070,

$y(i+25) = y(i+3)+y(i+2) +y(i+1) +y(i)$ *modulo 2, i=0,…,* 4070.

The definition of the *n*:th code word follows (the left most index correspond to the chip transmitted first in each slot):

$C_{RACH,n} = <x_n(0)+y(0), x_n(1)+y(1), \ldots,x_n(4095)+y(4095) >$,

All sums of symbols are taken modulo 2.

4.3.3.5 Scrambling code for the message part

In addition to spreading, the message part is also subject to scrambling with a 10 ms complex code. The scrambling code is cell-specific and has a one-to-one correspondence to the ~~spreading~~ scrambling code used for the preamble part.

The scrambling codes used are formed from the continuation of the ~~same~~ sequences x_n and *y* used for the preamble scrambling code and described in 4.3.3.1. Specifically, the values $x_n(4096)$, $x_n(4097)$,…,$x_n(4295)$ and $y(4096)$,$y(4097)$,…,$y(4295)$ are generated according to the recursive relations in 4.3.3.1 and used to form the *n*th constituent codes, $c_{1,n}$, and $c_{2,n}$ (the left most index corresponds to the first chip scrambled in the message):

$c_{1,n}= <x_n(4096)+y(4096), x_n(4097)+y(4097), \ldots,x_n(42495)+y(42495) >$,

$c_{2,n} = \langle x_n(M+4096)+y(M+4096), x_n(M+4097)+y(M+4097), \ldots, x_n(M+42495)+y(M+42495) \rangle$,

where *M* is defined in Table 3. The scrambling code for the message part is then

$C_{MSG,n} = c_{1,n} (w_0 + j\, c'_{2,n} w_1)$

where w_0 and w_1 are defined in 4.3.2.1 and $c'_{2,n}$ is a decimated version of $c_{2,n}$ as described in 4.3.2.1.

set of codes as is used for the other dedicated uplink channels when the long scrambling codes are used for these channels. The first 256 of the long scrambling codes are used for the random access channel. The phases 4096..42496 of the codes are used for the message part (phases 0..4095 of c_1 are used in preamble spreading) with the chip rate of 3.84 Mchips/s.