ПРОГРАММА КУРСА

Курс состоит из двух частей: теория и программирование. В зависимости от желаний и способностей студента при усвоении и исполнении большей или меньшей части предлагаемого курса возможно получение различных оценок.

ТЕОРИЯ

1. Общее описание системы UMTS.
2. Формирование нисходящих сигналов системы UMTS для режима FDD.
3. Формирование восходящих сигналов системы UMTS для режима FDD.
4. Коды, используемые в UMTS: первичная и вторичные ПСП, каналообразующие коды, различные скремблирующие последовательности.
5. Кадровая структура основных нисходящих каналов.
6. Кадровая структура основных восходящих каналов.
7. Канальное кодирование.
8. Rake-приемник для CDMA систем.
9. Принцип организации и передачи системной информации.

ПРОГРАММИРОВАНИЕ

ОБЩИЕ ФУНКЦИИ

1. Формирование первичной синхропоследовательности.
2. Формирование набора вторичных синхропоследовательностей (включая формирование матрицы Адамара).
3. Формирование таблицы номеров вторичных синхропоследовательностей в зависимости от номера скремблирующей группы
4. Формирование каналообразующего кода.
5. Формирование скремблирующей последовательности.
6. Проверка целостности пакета (деление полиномов).
7. Декодер Витерби.
8. Деперемежитель №1.
9. Деперемежитель №2.

ОСНОВНАЯ ОБРАБОТКА СИГНАЛОВ

0. Общая программа обработки сигнала.
1. Согласованная фильтрация сигнала.
2. Слотовая синхронизация по ПСП.
3. Кадровая синхронизация по ВСП.
4. Определение номера скремблирующей последовательности.
5. Вычисление rake-шаблона.
6. Однолучевая демодуляция общего канала управления.
7. Канальное декодирование вещательного канала.

ДОПОЛНИТЕЛЬНАЯ ОБРАБОТКА СИГНАЛОВ

1. Многолучевая демодуляция общего канала управления.
2. Синтаксический анализ системной информации.

ОЦЕНКА УСВОЕНИЯ МАТЕРИАЛА И ВЫПОЛНЕНИЯ ЗАДАНИЙ

«3» Для возможности получения оценки «удовлетворительно» необходимо уметь блочно (без подробностей) объяснять принципы формирования и обработки сигналов,

реализовать все общие функции и функции основной обработки сигналов, уметь объяснять алгоритмы функций блочно.

«4» Для возможности получения оценки «хорошо» дополнительно к предыдущему необходимо уметь подробно объяснять принципы формирования и обработки сигналов, реализовать многолучевую демодуляцию для общего канала управления, уметь подробно (построчно) объяснять алгоритмы всех функций.

«5» Для возможности получения оценки «отлично» дополнительно к предыдущему необходимо произвести синтаксический анализ системной информации.

ФОРМАЛЬНОЕ ОПИСАНИЕ ФУНКЦИЙ

ОБЩИЕ ФУНКЦИИ

1. Формирование первичной синхропоследовательности – функция без входных переменных, генерирующая комплексную первичную синхропоследовательность. Первые строки функции должны быть такими:

```
function PSC = Generate_Primary_Synchronisation_Code
% Функция генерирует первичную синхропоследовательность.
%
% Выходные переменные:
%   PSC - массив-строка длиной 256 элементов, содержащий комплексные
%         значения первичной синхропоследовательности.
```

2. Формирование набора вторичных синхропоследовательностей – функция без входных переменных, генерирующая набор из 16-и комплексных вторичных синхропоследовательностей. Внутри функции должна быть реализована функция генерирования матрицы Адамара заданного размера. Первые строки функций должны быть такими:

```
function SSC = Generate_Secondary_Synchronisation_Codes
% Функция генерирует набор из 16-и вторичных синхропоследовательностей.
%
% Выходные переменные:
%   SSC - массив размером 16*256 элементов, в строках которого находятся
%         комплексные значения вторичных синхропоследовательностей.

function H = Generate_Hadamard_Matrix(Rang)
% Функция генерирует матрицу Адамара заданного размера.
%
% Входные переменные:
%   Rang - параметр для расчета размера матрицы Адамара.
%
% Выходные переменные:
%   H - квадратная матрица Адамара размерностью 2^Rang * 2^Rang.
```

3. Формирование таблицы номеров вторичных синхропоследовательностей в зависимости от номера скремблирующей группы – текст данной функции представлен ниже:

```
function SG = Generate_Scrambling_Groups_Table
% Формирует таблицу номеров вторичных синхропоследовательностей (1..16) в
% зависимости от номера скремблирующей группы (0..63) и номера слота в
% кадре (0..14). Таким образом, SG - массив размером 64*15 чисел из
% диапазона (1..16).

% Слот: 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14
SG =   [1, 1, 2, 8, 9,10,15, 8,10,16, 2, 7,15, 7,16; ... % Скр. группа №0
        1, 1, 5,16, 7, 3,14,16, 3,10, 5,12,14,12,10; ... % Скр. группа №1
        1, 2, 1,15, 5, 5,12,16, 6,11, 2,16,11,15,12; ... % Скр. группа №2
        1, 2, 3, 1, 8, 6, 5, 2, 5, 8, 4, 4, 6, 3, 7; ... % Скр. группа №3
        1, 2,16, 6, 6,11,15, 5,12, 1,15,12,16,11, 2; ... % Скр. группа №4
        1, 3, 4, 7, 4, 1, 5, 5, 3, 6, 2, 8, 7, 6, 8; ... % Скр. группа №5
        1, 4,11, 3, 4,10, 9, 2,11, 2,10,12,12, 9, 3; ... % Скр. группа №6
        1, 5, 6, 6,14, 9,10, 2,13, 9, 2, 5,14, 1,13; ... % Скр. группа №7
        1, 6,10,10, 4,11, 7,13,16,11,13, 6, 4, 1,16; ... % Скр. группа №8
        1, 6,13, 2,14, 2, 6, 5, 5,13,10, 9, 1,14,10; ... % Скр. группа №9
        1, 7, 8, 5, 7, 2, 4, 3, 8, 3, 2, 6, 6, 4, 5; ... % Скр. группа №10
        1, 7,10, 9,16, 7, 9,15, 1, 8,16, 8,15, 2, 2; ... % Скр. группа №11
        1, 8,12, 9, 9, 4,13,16, 5, 1,13, 5,12, 4, 8; ... % Скр. группа №12
        1, 8,14,10,14, 1,15,15, 8, 5,11, 4,10, 5, 4; ... % Скр. группа №13
        1, 9, 2,15,15,16,10, 7, 8, 1,10, 8, 2,16, 9; ... % Скр. группа №14
        1, 9,15, 6,16, 2,13,14,10,11, 7, 4, 5,12, 3; ... % Скр. группа №15
```

```
 1,10, 9,11,15, 7, 6, 4,16, 5, 2,12,13, 3,14; ... % Скр. группа №16
 1,11,14, 4,13, 2, 9,10,12,16, 8, 5, 3,15, 6; ... % Скр. группа №17
 1,12,12,13,14, 7, 2, 8,14, 2, 1,13,11, 8,11; ... % Скр. группа №18
 1,12,15, 5, 4,14, 3,16, 7, 8, 6, 2,10,11,13; ... % Скр. группа №19
 1,15, 4, 3, 7, 6,10,13,12, 5,14,16, 8, 2,11; ... % Скр. группа №20
 1,16, 3,12,11, 9,13, 5, 8, 2,14, 7, 4,10,15; ... % Скр. группа №21
 2, 2, 5,10,16,11, 3,10,11, 8, 5,13, 3,13, 8; ... % Скр. группа №22
 2, 2,12, 3,15, 5, 8, 3, 5,14,12, 9, 8, 9,14; ... % Скр. группа №23
 2, 3, 6,16,12,16, 3,13,13, 6, 7, 9, 2,12, 7; ... % Скр. группа №24
 2, 3, 8, 2, 9,15,14, 3,14, 9, 5, 5,15, 8,12; ... % Скр. группа №25
 2, 4, 7, 9, 5, 4, 9,11, 2,14, 5,14,11,16,16; ... % Скр. группа №26
 2, 4,13,12,12, 7,15,10, 5, 2,15, 5,13, 7, 4; ... % Скр. группа №27
 2, 5, 9, 9, 3,12, 8,14,15,12,14, 5, 3, 2,15; ... % Скр. группа №28
 2, 5,11, 7, 2,11, 9, 4,16, 7,16, 9,14,14, 4; ... % Скр. группа №29
 2, 6, 2,13, 3, 3,12, 9, 7,16, 6, 9,16,13,12; ... % Скр. группа №30
 2, 6, 9, 7, 7,16,13, 3,12, 2,13,12, 9,16, 6; ... % Скр. группа №31
 2, 7,12,15, 2,12, 4,10,13,15,13, 4, 5, 5,10; ... % Скр. группа №32
 2, 7,14,16, 5, 9, 2, 9,16,11,11, 5, 7, 4,14; ... % Скр. группа №33
 2, 8, 5,12, 5, 2,14,14, 8,15, 3, 9,12,15, 9; ... % Скр. группа №34
 2, 9,13, 4, 2,13, 8,11, 6, 4, 6, 8,15,15,11; ... % Скр. группа №35
 2,10, 3, 2,13,16, 8,10, 8,13,11,11,16, 3, 5; ... % Скр. группа №36
 2,11,15, 3,11, 6,14,10,15,10, 6, 7, 7,14, 3; ... % Скр. группа №37
 2,16, 4, 5,16,14, 7,11, 4,11,14, 9, 9, 7, 5; ... % Скр. группа №38
 3, 3, 4, 6,11,12,13, 6,12,14, 4, 5,13, 5,14; ... % Скр. группа №39
 3, 3, 6, 5,16, 9,15, 5, 9,10, 6, 4,15, 4,10; ... % Скр. группа №40
 3, 4, 5,14, 4, 6,12,13, 5,13, 6,11,11,12,14; ... % Скр. группа №41
 3, 4, 9,16,10, 4,16,15, 3, 5,10, 5,15, 6, 6; ... % Скр. группа №42
 3, 4,16,10, 5,10, 4, 9, 9,16,15, 6, 3, 5,15; ... % Скр. группа №43
 3, 5,12,11,14, 5,11,13, 3, 6,14, 6,13, 4, 4; ... % Скр. группа №44
 3, 6, 4,10, 6, 5, 9,15, 4,15, 5,16,16, 9,10; ... % Скр. группа №45
 3, 7, 8, 8,16,11,12, 4,15,11, 4, 7,16, 3,15; ... % Скр. группа №46
 3, 7,16,11, 4,15, 3,15,11,12,12, 4, 7, 8,16; ... % Скр. группа №47
 3, 8, 7,15, 4, 8,15,12, 3,16, 4,16,12,11,11; ... % Скр. группа №48
 3, 8,15, 4,16, 4, 8, 7, 7,15,12,11, 3,16,12; ... % Скр. группа №49
 3,10,10,15,16, 5, 4, 6,16, 4, 3,15, 9, 6, 9; ... % Скр. группа №50
 3,13,11, 5, 4,12, 4,11, 6, 6, 5, 3,14,13,12; ... % Скр. группа №51
 3,14, 7, 9,14,10,13, 8, 7, 8,10, 4, 4,13, 9; ... % Скр. группа №52
 5, 5, 8,14,16,13, 6,14,13, 7, 8,15, 6,15, 7; ... % Скр. группа №53
 5, 6,11, 7,10, 8, 5, 8, 7,12,12,10, 6, 9,11; ... % Скр. группа №54
 5, 6,13, 8,13, 5, 7, 7, 6,16,14,15, 8,16,15; ... % Скр. группа №55
 5, 7, 9,10, 7,11, 6,12, 9,12,11, 8, 8, 6,10; ... % Скр. группа №56
 5, 9, 6, 8,10, 9, 8,12, 5,11,10,11,12, 7, 7; ... % Скр. группа №57
 5,10,10,12, 8,11, 9, 7, 8, 9, 5,12, 6, 7, 6; ... % Скр. группа №58
 5,10,12, 6, 5,12, 8, 9, 7, 6, 7, 8,11,11, 9; ... % Скр. группа №59
 5,13,15,15,14, 8, 6, 7,16, 8, 7,13,14, 5,16; ... % Скр. группа №60
 9,10,13,10,11,15,15, 9,16,12,14,13,16,14,11; ... % Скр. группа №61
 9,11,12,15,12, 9,13,13,11,14,10,16,15,14,16; ... % Скр. группа №62
 9,12,10,15,13,14, 9,14,15,11,11,13,12,16,10];    % Скр. группа №63
```

4. Формирование каналообразующего кода – функция генерирует каналообразующий код. Входными переменными являются значение коэффициента расширения и номер кода. Первые строки функции должны быть такими:

```
function Ch = Generate_Channelisation_Code(SF, k)
% Функция генерирует каналообразующий код по заданным значениям
% коэффициента расширения и номера кода.
%
% Входные переменные:
%   SF - коэффициент расширения; может принимать значения, равные 2^n,
%        где n – положительное целое;
%   k  - номер кода, может принимать значения k = 0, ..., SF-1.
%
% Выходные переменные:
```

```
%   Ch - строка длиной SF элементов, содержащая значения каналообразующего
%        кода.
```

5. Формирование скремблирующей последовательности – функция генерирует скремблирующую последовательность по заданному номеру. Первые строки функции должны быть такими:

```
function Sn = Generate_Scrambling_Code(n)
% Функция генерирует скремблирующую последовательность по заданному номеру.
%
% Входные переменные:
%   n - номер скремблирующей последовательности, n = 0, ..., 511.
%
% Выходные переменные:
%   Sn – массив-строка длиной 38400 элементов, содержащая комплексные
%        значения скремблирующей последовательности.
```

6. Проверка целостности пакета – функция выполняет проверку целостности пакета путем сравнения остатка от деления полиномов с нулем. Входными переменными являются массив-строка (в строке содержатся как полезные данные, так и биты CRC) и размер блока CRC. Выходными переменными являются флаг успешной проверки целостности пакета и массив-строка полезных данных. Внутри функции должна быть реализована функция деления полиномов. Первые строки функций должны быть такими:

```
function [Flag_isOk, Data] = Check_CRC(InVect, CRC_Size)
% Функция выполняет проверку целостности пакета по  заданному значению
% размера блока CRC.
%
% Входные переменные:
%   InVect    – массив-строка, длина которого д.б. больше CRC_Size,
%               первыми следуют полезные биты, далее биты CRC,
%               записанные в обратном порядке;
%   CRC_Size  – размер блока CRC, м.б. равен {24, 16, 12, 8};
%   Flag_isOk – флаг проверки целостности пакета, Flag_isOk = true,
%               если блок признан безошибочным, Flag_isOk = false
%               в противоположном случае.
%
% Выходные переменные:
%   Data - если Flag_isOk = true, то Data - массив-строка полезных данных,
%          иначе Data = [].

function [Quotient, Remainder] = Polynom_Division(Dividend, Denominator)
% Функция выполняет деление полиномов при этом
% Dividend = Quotient * Denominator + Remainder.
%
% Все переменные – массивы-строки, содержащие значения коэффициентов
% стоящих при степенях полиномов, при этом первый по порядку элемент
% соответствует коэффициенту при старшей степени полинома.
%
% Входные переменные:
%   Dividend    - делимый полином;
%   Denominator – полином-делитель;
%
% Выходные переменные:
%   Quotient    - полином-частное от деления;
%   Remainder   - полином-остаток от деления.
%
% Пример: Dividend    = x^4 + x^3 + 1 = [1, 1, 0, 0, 1]
%         Denominator =       x^2 + 1 =       [1, 0, 1]
%         Quotient    =   x^2 + x + 1 =       [1, 1, 1]
%         Remainder   =             x =          [1, 0]
```

7. Декодер Витерби – функция выполняет декодирование блока полезных данных, кодированного сверточным кодером с учетом нулевого начального и конечного (благодаря добавлению концевых бит) состояния регистра кодера. Входными переменными являются массив-строка кодированных данных и тип кодера. Выходной переменной является массив-строка декодированных данных. В программе необходимо использовать встроенные функции poly2trellis и vitdec. Первые строки функции должны быть такими:

```
function Decoded_Vect = Convolutional_Decoder(Coded_Vect, Flag_isHalf)
% Функция выполняет декодирование по алгоритму Витерби блока данных,
% кодированного сверточным кодом.
%
% Входные переменные:
%   Coded_Vect  - массив-строка кодированных данных;
%   Flag_isHalf - флаг указывающий на то, какая скорость кодера
%                 была использована при получении Coded_Vect:
%                 Flag_isHalf = true  - 1/2,
%                 Flag_isHalf = false - 1/3.
%
% Выходные переменные:
%   Decoded_Vect - массив-строка декодированных данных.
```

8. Деперемежитель №1 – выполняет процедуру обратную относительно процедуры первого перемежения. Входными переменными являются длительность интервала передачи блока данных (в единицах кадров) и блок перемеженных данных. Выходной переменной является блок деперемеженных данных. Первые строки функции должны быть такими:

```
function Deinterleaved_Vect = First_DeInterleaver(Interleaved_Vect, TTI)
% Функция выполняет процедуру обратную относительно процедуры первого
% перемежения.
%
% Входные переменные:
%   Interleaved_Vect - массив-строка перемеженных данных;
%   TTI              - длительность интервала передачи блока данных
%                      в единицах кадров, м.б. равна 1, 2, 4, 8.
%
% Выходные переменные:
%   Deinterleaved_Vect - массив-строка деперемеженных данных.
```

9. Деперемежитель №2 – выполняет процедуру обратную относительно процедуры второго перемежения. Входной переменной является блок перемеженных данных. Выходной переменной является блок деперемеженных данных. Первые строки функции должны быть такими:

```
function Deinterleaved_Vect = Second_DeInterleaver(Interleaved_Vect)
% Функция выполняет процедуру обратную относительно процедуры второго
% перемежения.
%
% Входные переменные:
%   Interleaved_Vect - массив-строка перемеженных данных.
%
% Выходные переменные:
%   Deinterleaved_Vect - массив-строка деперемеженных данных.
```

ОСНОВНАЯ ОБРАБОТКА СИГНАЛОВ

0. Общая программа основной обработки сигнала. Далее представлен код скрипта, который при наличии корректно-работающих функций осуществляет необходимую основную обработку сигнала.

```
% Очистка динамической памяти и Command Window
    clc;
    clear;
```

```
% Загрузка массива Signal с записью сигнала
    load('Beeline'); % В кавычках указывается имя файла из
        % которого будет считана запись сигнала

% Согласованная фильтрация сигнала
    df = 0; % Пока что нет оснований выбрать другое значение частотной
            % отстройки
    FSignal = Matched_Filter(Signal, 0);

% Слотовая синхронизация - поиск базовых станций (БС)
    Slots_Offsets = Slot_Synchronization(FSignal, true);
% Для каждой найденной БС проводим следующие процедуры обработки
    if ~isempty(Slots_Offsets)
        % Создадим переменную для хранения транспортных блоков вещательного
        % канала найденных БС
            BCCHs = cell(length(Slots_Offsets), 1);
        for k = 1:length(Slots_Offsets) % Для каждой БС
            % Кадровая синхронизация
                [Frame_Offset, SG] = Frame_Synchronization(FSignal, ...
                    Slots_Offsets(1, k), true);
            % Определение номера скремблирующей последовательности
                SC_Num = Scrambling_Code_Determination(FSignal, ...
                    Frame_Offset, SG, true);
            % Построение rake-шаблона
                Rake_Pattern = Rake_Pattern_Calculation(Signal, ...
                    FSignal, Frame_Offset, SC_Num, true);
            % Демодуляция вещательного канала
                PCCPCH_Bits = One_Ray_PCCPCH_Demodulation(Signal, ...
                    Rake_Pattern, Frame_Offset, SC_Num, true);
            % Декодирование транспортных блоков вещательного канала
                [Flag_isOk, BCCH] = Decode_BCCH(PCCPCH_Bits);
                display(Flag_isOk);
            % Сохранение полученных данных
                BCCHs{k, 1} = BCCH;
        end
    end
```

1. Согласованная фильтрация сигнала – функция производит вычисление импульсного отклика фильтра, согласованного с формирующим фильтром системы UMTS, подстройку по частоте и согласованную фильтрацию сигнала, который представлен массивом комплексных отсчетов, записанных с частотой дискретизации, равной удвоенной символьной скорости. Входными переменными являются массив данных, содержащий комплексные отсчеты сигнала (далее исходный сигнал), и значение подстройки по частоте в единицах символьной скорости. Выходной переменной является массив значений сигнала с выхода согласованного фильтра (далее сигнал с выхода согласованного фильтра). Первые строки функции должны быть такими:

```
function FSignal = Matched_Filter(Signal, df)
% Функция производит фильтрацию сигнала фильтром, согласованным с
% формирующим.
%
% Входные переменные:
%   Signal - комплексный массив, содержащий отсчеты исходного сигнала;
%   df     - частотная отстройка в единицах символьной скорости.
%
% Выходные переменные:
%   FSignal - комплексный массив, содержащий отсчеты фильтрованного
%   сигнала.
```

2. Слотовая синхронизация по ПСП – функция выполняет поиск сигналов базовых станций по первичной синхропоследовательности. Входными переменными являются сигнал с

выхода согласованного фильтра и флаг необходимости прорисовки корреляционной кривой ПСП. Выходной переменной является массив значений сдвигов до начала слотов найденных сигналов базовых станций относительно начала массива фильтрованного сигнала в единицах отсчетов. Первые строки функции должны быть такими:

```
function Slots_Offsets = Slot_Synchronization(FSignal, Flag_Draw)
% Функция выполняет процедуру слотовой синхронизации.
%
% Входные переменные:
%   FSignal   - комплексный массив, содержащий отсчеты фильтрованного
%               сигнала;
%   Flag_Draw - флаг необходимости прорисовки корреляционной
%               кривой, Flag_Draw = true указывает на
%               необходимость прорисовки.
%
% Выходные переменные:
%   Slots_Offsets - массив значений сдвигов в FSignal до начала слотов
%                   найденных сигналов базовых станций.
```

3. Кадровая синхронизация по ВСП – функция выполняет кадровую синхронизацию и определяет номер скремблирующей группы. Входными переменными являются сигнал с выхода согласованного фильтра, значение сдвига до начала слота и флаг необходимости прорисовки корреляционных кривых ВСП для каждого слота. Выходными переменными являются значение сдвига до начала кадра относительно начала массива фильтрованного сигнала в единицах отсчетов и значение номера скремблирующей группы. Первые строки функции должны быть такими:

```
function [Frame_Offset, SG] = Frame_Synchronization(FSignal, ...
    Slot_Offset, Flag_Draw)
% Функция выполняет процедуру кадровой синхронизации.
%
% Входные переменные:
%   FSignal     - комплексный массив, содержащий отсчеты фильтрованного
%                 сигнала;
%   Slot_Offset - значение сдвига в FSignal до начала слота сигнала
%                 базовой станций;
%   Flag_Draw   - флаг необходимости прорисовки корреляционной
%                 кривой, Flag_Draw = true указывает на
%                 необходимость прорисовки;
%
% Выходные переменные:
%   Frame_Offset - значение сдвига в FSignal до начала кадра;
%   SG           - номер скремблирующей группы.
```

4. Определение номера скремблирующей последовательности – функция производит определение номера скремблирующей последовательности (СП) по общему пилотному каналу. Входными переменными являются сигнал с выхода согласованного фильтра, значение сдвига до начала кадра, номер скремблирующей группы и флаг необходимости прорисовки корреляционной кривой СП. Выходной переменной является значение номера СП. Первые строки функции должны быть такими:

```
function SC_Num = Scrambling_Code_Determination(FSignal, ...
    Frame_Offset, SG, Flag_Draw)
% Функция выполняет процедуру определения номера скремблирующей
% последовательности.
%
% Входные переменные:
%   FSignal - комплексный массив, содержащий отсчеты фильтрованного
%             сигнала;
%   SG      - номер скремблирующей группы.
%
% Выходные переменные:
```

```
%   SC_Num - номер скремблирующей последовательности.
```

5. Вычисление rake-шаблона – функция производит оценку частотной отстройки сигналов лучей, пришедших в разные моменты времени в окрестности синхронизации с сигналом БС, и вычисляет значения корреляционной функции пилотного канала на одном кадре для этих же сигналов. Входными переменными являются исходный сигнал и сигнал с выхода согласованного фильтра, значение сдвига до начала кадра, значение номера СП и флаг необходимости прорисовки визуализационных кривых (например, зависимости значений КФ пилотного канала от задержки луча). Выходной переменной является структура rake-шаблона, содержащая два поля: значения КФ пилотного канала для лучей с разной задержкой и значения оценок частотной отстройки тех же лучей (в долях символьной скорости). Первые строки функции должны быть такими:

```
function Rake_Pattern = Rake_Pattern_Calculation(Signal, FSignal, ...
    Frame_Offset, SC_Num, Flag_Draw)
% Функция выполняет процедуру построения rake-шаблона.
%
% Входные переменные:
%   Signal       - комплексный массив, содержащий отсчеты исходного
%                  сигнала;
%   FSignal      - комплексный массив, содержащий отсчеты фильтрованного
%                  сигнала;
%   Frame_Offset - значение сдвига в FSignal до начала кадра;
%   SC_Num       - номер скремблирующей последовательности;
%   Flag_Draw    - флаг необходимости прорисовки корреляционной
%                  кривой, Flag_Draw = true указывает на
%                  необходимость прорисовки.
%
% Выходные переменные:
%   Rake_Pattern - rake-шаблона, структура, содержащая два поля данных:
%     Correl     - значения КФ пилотного канала       для разных лучей;
%     dfs        - значения оценок частотных отстроек для разных лучей.
```

6. Однолучевая демодуляция общего канала управления – функция выполняет демодуляцию общего канала управления по основному лучу с использованием эквалайзера, настроенного на пилотный канал. Входными переменными являются исходный сигнал, структура rake-шаблона, значение сдвига до начала кадра, номер скремблирующей последовательности и флаг необходимости прорисовки комплексных отсчетов общего канала управления. Выходной переменной является массив-строка демодулированных символов общего канала управления. Первые строки функции должны быть такими:

```
function PCCPCH_Bits = One_Ray_PCCPCH_Demodulation(Signal, ...
    Rake_Pattern, Frame_Offset, SC_Num, Flag_Draw)
% Функция выполняет однолучевую демодуляцию всех кадров вещательного канала
% имеющихся в Signal.
%
% Входные переменные:
%   Signal       - комплексный массив, содержащий отсчеты исходного
%                  сигнала;
%   Rake_Pattern - rake-шаблона, структура, содержащая два поля данных:
%     Correl     - значения КФ пилотного канала       для разных лучей;
%     dfs        - значения оценок частотных отстроек для разных лучей;
%   Frame_Offset - значение сдвига в FSignal до начала кадра;
%   SC_Num       - номер скремблирующей последовательности;
%   Flag_Draw    - флаг необходимости прорисовки корреляционной
%                  кривой, Flag_Draw = true указывает на
%                  необходимость прорисовки.
%
% Выходные переменные:
%   PCCPCH_Bits - массив-строка (длина кратна 270), содержащий значения
%                 бит всех кадров канала PCCPCH, считанных из Signal.
```

7. Канальное декодирование вещательного канала – функция выполняет канальное декодирование вещательного канала до транспортных блоков. Входной переменной является массив-строка демодулированных символов общего канала управления. Выходными переменными являются флаг, указывающий на успешное декодирование хотя бы одного транспортного блока вещательного канала, и сами транспортные блоки. Первые строки функции должны быть такими:

```
function [Flag_isOk, BCCH] = Decode_BCCH(Coded_BCCH)
% Функция выполняет по-2-ух-кадровое декодирование данных вещательного
% канала.
%
% Входные переменные:
%   Coded_BCCH - массив-строка (длина кратна 270), содержащий значения
%                бит всех считанных кадров канала BCCH.
%
% Выходные переменные:
%   BCCH      – массив с количеством строк, равным количеству декодирован-
%               ых транспортных блоков BCCH, и количеством столбцов 246;
%   Flag_isOk - указывает на то, был ли успешно декодирован хотя бы один
%               транспортный блок BCCH.
```

ДОПОЛНИТЕЛЬНАЯ ОБРАБОТКА СИГНАЛОВ

1. Многолучевая демодуляция общего канала управления – функция выполняет демодуляцию общего канала управления по нескольким лучам с использованием эквалайзера, настроенного на пилотный канал и данных rake-шаблона. Формальное описание функции совпадает со случаем однолучевой демодуляции. Первые строки функции должны быть такими:

```
function PCCPCH_Bits = Multi_Rays_PCCPCH_Demodulation(Signal, ...
    Rake_Pattern, Frame_Offset, SC_Num, Flag_Draw)
% Функция выполняет однолучевую демодуляцию всех кадров вещательного канала
% имеющихся в Signal.
%
% Входные переменные:
%   Signal       - комплексный массив, содержащий отсчеты исходного
%                  сигнала;
%   Rake_Pattern - rake-шаблона, структура, содержащая два поля данных:
%     Correl     - значения КФ пилотного канала       для разных лучей;
%     dfs        - значения оценок частотных отстроек для разных лучей;
%   Frame_Offset – значение сдвига в FSignal до начала кадра;
%   SC_Num       - номер скремблирующей последовательности;
%   Flag_Draw    - флаг необходимости прорисовки корреляционной
%                  кривой, Flag_Draw = true указывает на
%                  необходимость прорисовки.
%
% Выходные переменные:
%   PCCPCH_Bits - массив-строка (длина кратна 270), содержащий значения
%                 бит всех кадров канала PCCPCH, считанных из Signal.
```

2. Синтаксический анализ системной информации – функция выполняет выделение полезных данных из транспортных блоков вещательного канала. Основными выделяемыми переменными являются код страны и код оператора. Остальные переменные устанавливаются по дополнительному согласованию с преподавателем. Входной переменной является массив транспортных блоков вещательного канала. Выходными переменными являются флаг, указывающий на успешный поиск основных переменных, основные переменные – код оператора и код страны и дополнительные переменные. Первые строки функции должны быть такими:

```
function [Flag_isOk, MCC, MNC, Other] = Parse_BCCH(BCCH)
```

```
% Функция выполняет синтаксический анализ полученных данных вещательного
% канала. Основными извлекаемыми параметрами являются код страны MCC и код
% оператора MNC.
%
% Входные переменные:
%   BCCH – массив с количеством строк, равным количеству декодированых
%          транспортных блоков BCCH, и количеством столбцов 246.
%
% Выходные переменные:
%   Flag_isOk - указывает на то, удалось ли считать параметры MCC, MNC;
%   MCC, MNC  - код страны и код оператора;
%   Other     - другие параметры вещательного канала по согласованию с
%               преподавателем.
```

УКАЗАНИЯ

Первичная и вторичная синхропоследовательности

Для определения первичной синхропоследовательности C_{PSC} (ПСП) введем сначала малую 16-ти элементную последовательность a:

$$a = <x_1, x_2, x_3, \ldots, x_{16}> = <1, 1, 1, 1, 1, 1, -1, -1, 1, -1, 1, -1, 1, -1, -1, 1>.$$

Тогда построение ПСП может быть осуществлено так:

$$C_{PSC} = (1 + j) \times <a, a, a, -a, -a, a, -a, -a, a, a, a, -a, a, -a, a, a>;$$

причем крайний левый элемент последовательности соответствует чипу, передаваемому первым во времени.

16 вторичных синхропоследовательностей $C_{SSC,\,k}$ (ВСП) также как и ПСП являются комплексными последовательностями с совпадающими мнимой и вещественной частями. Их построение основано на поэлементном умножении определенной последовательности Адамара и последовательности z, определяемой так:

$$z = <b, b, b, -b, b, b, -b, -b, b, -b, b, -b, -b, -b, -b, -b>, \text{ где}$$

$$b = <x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, -x_9, -x_{10}, -x_{11}, -x_{12}, -x_{13}, -x_{14}, -x_{15}, -x_{16}>$$

и x_1, x_2 , …, x_{15}, x_{16}, такие же как при определении последовательности a для ПСП. Последовательности Адамара h_n являются рядами матрицы H_8, получаемой рекурсией:

$$H_0 = (1),$$

$$H_k = \begin{pmatrix} H_{k-1} & H_{k-1} \\ H_{k-1} & -H_{k-1} \end{pmatrix}, \quad k \geq 1.$$

Ряды нумеруются сверху вниз начиная со значения 0 (последовательность только из единиц) и заканчивая значением 255. Обозначим через $h_n(i)$ и $z(i)$ i-ые элементы последовательностей h_n и z соответственно, где $i = 0, 1, 2, \ldots, 255$ и $i = 0$ соответствует крайнему левому элементу. Тогда k-ая ВСП $C_{SSC,\,k}$, $k = 1, 2, 3, \ldots, 16$ будет определяться так:

$$C_{SSC,\,k} = (1 + j) \times <h_m(0) \times z(0), h_m(1) \times z(1), h_m(2) \times z(2), \ldots, h_m(255) \times z(255)>,$$

где $m = 16 \times (k - 1)$ и крайний левый элемент последовательности соответствует чипу, передаваемому первым во времени.

В табл. 1 указано распределение ВСП в слотах одного кадра для различных кодовых скремблирующих групп.

Табл. 1. Распределение ВСП в слотах в зависимости от кодовой скремблирующей группы

кодовая скремблирующая группа	Номер слота														
	#0	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12	#13	#14
Группа 0	1	1	2	8	9	10	15	8	10	16	2	7	15	7	16
Группа 1	1	1	5	16	7	3	14	16	3	10	5	12	14	12	10
Группа 2	1	2	1	15	5	5	12	16	6	11	2	16	11	15	12
Группа 3	1	2	3	1	8	6	5	2	5	8	4	4	6	3	7
Группа 4	1	2	16	6	6	11	15	5	12	1	15	12	16	11	2
Группа 5	1	3	4	7	4	1	5	5	3	6	2	8	7	6	8
Группа 6	1	4	11	3	4	10	9	2	11	2	10	12	12	9	3
Группа 7	1	5	6	6	14	9	10	2	13	9	2	5	14	1	13
Группа 8	1	6	10	10	4	11	7	13	16	11	13	6	4	1	16
Группа 9	1	6	13	2	14	2	6	5	5	13	10	9	1	14	10
Группа 10	1	7	8	5	7	2	4	3	8	3	2	6	6	4	5
Группа 11	1	7	10	9	16	7	9	15	1	8	16	8	15	2	2
Группа 12	1	8	12	9	9	4	13	16	5	1	13	5	12	4	8
Группа 13	1	8	14	10	14	1	15	15	8	5	11	4	10	5	4

Группа 14	1	9	2	15	15	16	10	7	8	1	10	8	2	16	9
Группа 15	1	9	15	6	16	2	13	14	10	11	7	4	5	12	3
Группа 16	1	10	9	11	15	7	6	4	16	5	2	12	13	3	14
Группа 17	1	11	14	4	13	2	9	10	12	16	8	5	3	15	6
Группа 18	1	12	12	13	14	7	2	8	14	2	1	13	11	8	11
Группа 19	1	12	15	5	4	14	3	16	7	8	6	2	10	11	13
Группа 20	1	15	4	3	7	6	10	13	12	5	14	16	8	2	11
Группа 21	1	16	3	12	11	9	13	5	8	2	14	7	4	10	15
Группа 22	2	2	5	10	16	11	3	10	11	8	5	13	3	13	8
Группа 23	2	2	12	3	15	5	8	3	5	14	12	9	8	9	14
Группа 24	2	3	6	16	12	16	3	13	13	6	7	9	2	12	7
Группа 25	2	3	8	2	9	15	14	3	14	9	5	5	15	8	12
Группа 26	2	4	7	9	5	4	9	11	2	14	5	14	11	16	16
Группа 27	2	4	13	12	12	7	15	10	5	2	15	5	13	7	4
Группа 28	2	5	9	9	3	12	8	14	15	12	14	5	3	2	15
Группа 29	2	5	11	7	2	11	9	4	16	7	16	9	14	14	4
Группа 30	2	6	2	13	3	3	12	9	7	16	6	9	16	13	12
Группа 31	2	6	9	7	7	16	13	3	12	2	13	12	9	16	6
Группа 32	2	7	12	15	2	12	4	10	13	15	13	4	5	5	10
Группа 33	2	7	14	16	5	9	2	9	16	11	11	5	7	4	14
Группа 34	2	8	5	12	5	2	14	14	8	15	3	9	12	15	9
Группа 35	2	9	13	4	2	13	8	11	6	4	6	8	15	15	11
Группа 36	2	10	3	2	13	16	8	10	8	13	11	11	16	3	5
Группа 37	2	11	15	3	11	6	14	10	15	10	6	7	7	14	3
Группа 38	2	16	4	5	16	14	7	11	4	11	14	9	9	7	5
Группа 39	3	3	4	6	11	12	13	6	12	14	4	5	13	5	14
Группа 40	3	3	6	5	16	9	15	5	9	10	6	4	15	4	10
Группа 41	3	4	5	14	4	6	12	13	5	13	6	11	11	12	14
Группа 42	3	4	9	16	10	4	16	15	3	5	10	5	15	6	6
Группа 43	3	4	16	10	5	10	4	9	9	16	15	6	3	5	15
Группа 44	3	5	12	11	14	5	11	13	3	6	14	6	13	4	4
Группа 45	3	6	4	10	6	5	9	15	4	15	5	16	16	9	10
Группа 46	3	7	8	8	16	11	12	4	15	11	4	7	16	3	15
Группа 47	3	7	16	11	4	15	3	15	11	12	12	4	7	8	16
Группа 48	3	8	7	15	4	8	15	12	3	16	4	16	12	11	11
Группа 49	3	8	15	4	16	4	8	7	7	15	12	11	3	16	12
Группа 50	3	10	10	15	16	5	4	6	16	4	3	15	9	6	9
Группа 51	3	13	11	5	4	12	4	11	6	6	5	3	14	13	12
Группа 52	3	14	7	9	14	10	13	8	7	8	10	4	4	13	9
Группа 53	5	5	8	14	16	13	6	14	13	7	8	15	6	15	7
Группа 54	5	6	11	7	10	8	5	8	7	12	12	10	6	9	11
Группа 55	5	6	13	8	13	5	7	7	6	16	14	15	8	16	15
Группа 56	5	7	9	10	7	11	6	12	9	12	11	8	8	6	10
Группа 57	5	9	6	8	10	9	8	12	5	11	10	11	12	7	7
Группа 58	5	10	10	12	8	11	9	7	8	9	5	12	6	7	6
Группа 59	5	10	12	6	5	12	8	9	7	6	7	8	11	11	9
Группа 60	5	13	15	15	14	8	6	7	16	8	7	13	14	5	16
Группа 61	9	10	13	10	11	15	15	9	16	12	14	13	16	14	11
Группа 62	9	11	12	15	12	9	13	13	11	14	10	16	15	14	16
Группа 63	9	12	10	15	13	14	9	14	15	11	11	13	12	16	10

Каналообразующие коды

В качестве каналообразующих кодов используются ортогональные коды с переменным коэффициентом расширения. Их формирование на основе кодового дерева пояснено на рис. 1:

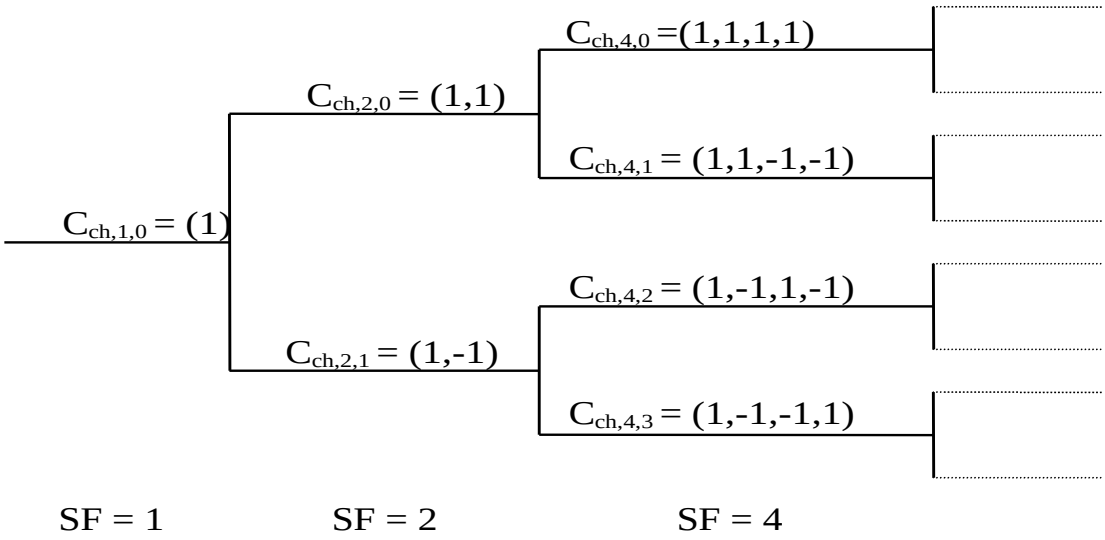


Рис. 1. Кодовое дерево ортогональных кодов с переменным коэффициентом расширения

На рис. 1 каждый каналообразующий код имеет уникальное название в формате $C_{ch,SF,k}$, где SF – коэффициент расширения кода и k – номер кода, $0 \le k \le SF - 1$. Каждый уровень кодового дерева на рис. 1 определяет каналообразующие коды длительности SF. Аналитическое правило генерирования каналообразующих кодов выглядит так:

$$C_{\text{ch},1,0} = 1,$$

$$\begin{bmatrix} C_{ch,2,0} \\ C_{ch,2,1} \end{bmatrix} = \begin{bmatrix} C_{ch,1,0} & C_{ch,1,0} \\ C_{ch,1,0} & -C_{ch,1,0} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix},$$

$$\begin{bmatrix} C_{ch,2^{(n+1)},0} \\ C_{ch,2^{(n+1)},1} \\ C_{ch,2^{(n+1)},2} \\ C_{ch,2^{(n+1)},3} \\ \vdots \\ C_{ch,2^{(n+1)},2^{(n+1)}-2} \\ C_{ch,2^{(n+1)},2^{(n+1)}-1} \end{bmatrix} = \begin{bmatrix} C_{ch,2^n,0} & C_{ch,2^n,0} \\ C_{ch,2^n,0} & -C_{ch,2^n,0} \\ C_{ch,2^n,1} & C_{ch,2^n,1} \\ C_{ch,2^n,1} & -C_{ch,2^n,1} \\ \vdots & \vdots \\ C_{ch,2^n,2^n-1} & C_{ch,2^n,2^n-1} \\ C_{ch,2^n,2^n-1} & -C_{ch,2^n,2^n-1} \end{bmatrix}.$$

Крайний левый элемент каждого каналообразующего кода соответствует чипу, передаваемому первым во времени.

Для общего пилот-канала используется каналообразующий код $C_{ch,256,0}$, а для первичного общего физического канала управления $C_{ch,256,1}$.

Скремблирующие коды

По описанному ниже алгоритму может быть сформировано $2^{18} - 1 = 262\,143$ различных скремблирующих кодов с номерами 0…262142. Тем не менее, не все коды используются в системе UMTS. Все множество скремблирующих кодов делится на 512 наборов, каждый из которых состоит из одного первичного и 15 вторичных скремблирующих кодов.

Номера первичных скремблирующих кодов удовлетворяют соотношению: $n = 16i$, где $i = 0 \ldots 511$. Вторичными скремблирующими кодами i-го набора являются коды с номерами $16i + k$, где $k = 1 \ldots 15$.

Множество первичных скремблирующих кодов делится на 64 кодовых скремблирующих группы, каждая из которых состоит из 8 первичных скремблирующих кодов. Причем j-я кодовая скремблирующая группа состоит из первичных скремблирующих кодов с номерами $16 \cdot 8j + 16k$, где $j = 0 \ldots 63$ и $k = 0 \ldots 7$.

Последовательность скремблирующего кода строится путем объединения двух вещественных последовательностей в одну комплексную. Скремблирующий код повторяется каждые 10 мсек, т.е. каждый кадр.

Введем порождающие вещественные последовательности x и y. Последовательность x строится на основе примитивного (в поле Галуа(2)) полинома $1 + q^7 + q^{18}$. Последовательность y строится на основе полинома $1 + q^5 + q^7 + q^{10} + q^{18}$.

Введем также последовательность z_n, зависящую от номера скремблирующего кода. Наконец, пусть $x(i)$, $y(i)$ и $z_n(i)$ обозначают i-ый элемент последовательности x, y и z_n, соответственно.

Правило построения x и y последовательностей следующее:

Начальные присвоения:

$x(0) = 1$, $x(1) = x(2) = \ldots = x(16) = x(17) = 0$,

$y(0) = y(1) = \ldots = y(16) = y(17) = 1$.

Рекурсивное определение последовательностей:

$x(i + 18) = x(i + 7) + x(i) \bmod 2$, $i = 0,\ldots,2^{18} - 20$,

$y(i + 18) = y(i + 10) + y(i + 7) + y(i + 5) + y(i) \bmod 2$, $i = 0,\ldots, 2^{18} - 20$.

Последовательность z_n, $n = 0, 1, 2, \ldots, 2^{18} - 2$, определяется так:

$z_n(i) = x((i + n) \bmod (2^{18} - 1)) + y(i) \bmod 2$, $i = 0, \ldots, 2^{18} - 2$.

Полученная последовательность далее преобразуется по следующему правилу:

$$Z_n(i) = \begin{cases} +1 & if\ z_n(i) = 0 \\ -1 & if\ z_n(i) = 1 \end{cases} \quad for \quad i = 0,1,\ldots,2^{18} - 2.$$

Наконец последовательность комплексного скремблирующего кода S_n получается так:

$S_n(i) = Z_n(i) + j\, Z_n((i + 131072) \bmod (2^{18} - 1))$, $i = 0, 1, \ldots, 38399$.

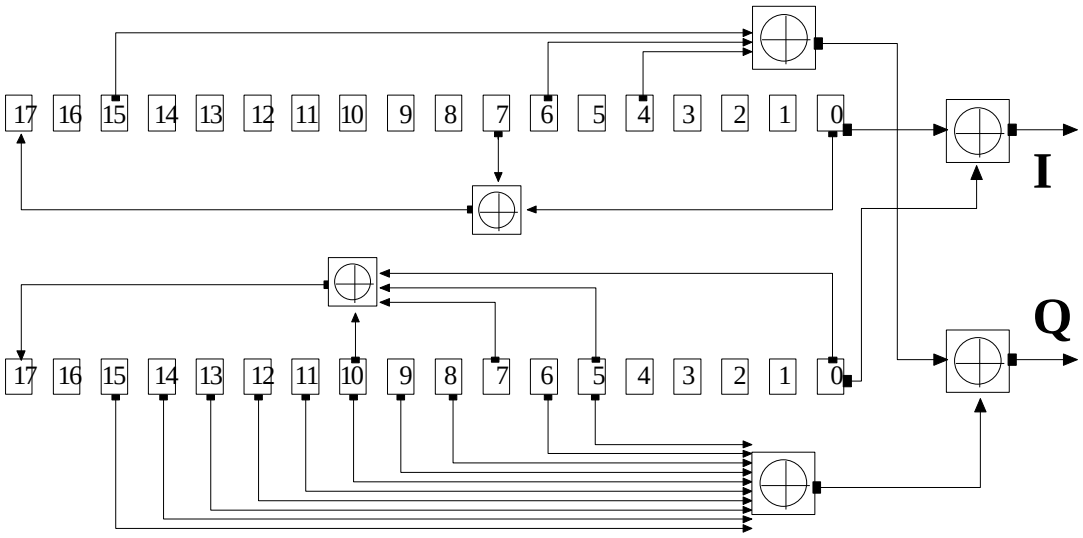


Рис. 2. Генератор скремблирующего кода

Добавление проверочных бит CRC

Для обнаружения ошибочных бит в UMTS используется CRC (Cyclic Redundancy Check). Количество проверочных бит может быть равным 24, 16, 12, 8 и 0, о чем сообщается с верхних уровней.

Для вычисления бит CRC используется весь транспортный блок. Для определения правила вычисления проверочных бит введем следующие обозначения. $a_1, a_2, \ldots, a_A$ – биты транспортного блока; $p_1, p_2, \ldots, p_L$ – проверочные биты CRC, где A – размер транспортного блока, L – количество проверочных бит CRC. Также введем в рассмотрение полиномы:

$$g_{CRC24}(D) = D^{24} + D^{23} + D^6 + D^5 + D + 1;$$
$$g_{CRC16}(D) = D^{16} + D^{12} + D^5 + 1;$$
$$g_{CRC12}(D) = D^{12} + D^{11} + D^3 + D^2 + D + 1;$$
$$g_{CRC8}(D) = D^8 + D^7 + D^4 + D^3 + D + 1.$$

В качестве бит CRC используются такие биты, что остаток от деления полинома

$$a_1 D^{A+L-1} + a_2 D^{A+L-2} + \ldots + a_A D^L + p_1 D^{L-1} + p_2 D^{L-2} + \ldots + p_{L-1} D^1 + p_L$$

на полином $g_{CRCL}(D)$ в поле Галуа GF(2) равен нулю.

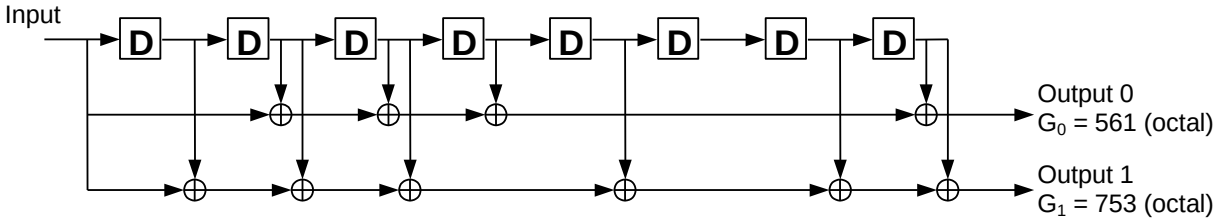
Обозначим через $b_1, b_2, \ldots, b_B$, где $B = A + L$, биты транспортного блока после добавления бит CRC. Тогда:

$b_k = a_k$ для $k = 1, 2, \ldots, A$;

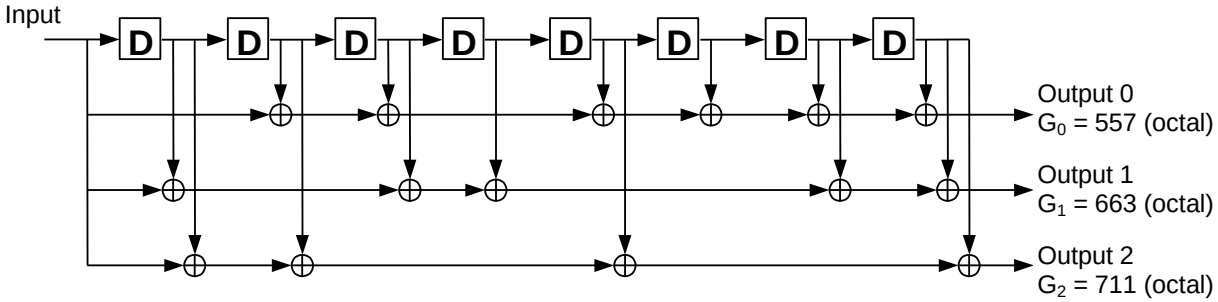
$b_k = p_{L+1-(k-A)}$ для $k = A + 1, A + 2, \ldots, A + L$.

Сверточный кодер

В качестве сверточного кодера в UMTS используются две схемы, обеспечивающие скорости кодирования 1/2 и 1/3 (рис. 3). Кодирование начинается с нулевого состояния регистра. При кодировании восемь нулевых бит добавляются в конец кодируемого сообщения, обеспечивая нулевое конечное состояние регистра. Таким образом, если длина кодируемого блока K, то длина кодированного блока равна $Y = 2K + 16$ либо $Y = 3K + 24$ в зависимости от кодовой скорости. При этом биты кодированного блока получаются путем поочередной записи значений с разных выводов кодера: выход0, выход1, выход0, выход1, …, либо выход0, выход1, выход2, выход0, выход1, выход2.



(a) Rate 1/2 convolutional coder



(b) Rate 1/3 convolutional coder

Рис. 3. Схемы сверточных кодеров, используемых в системе UMTS

Перемежитель №1

В качестве первого перемежителя используется блоковый перемежитель. Обозначим последовательность, поступающую на перемежитель, x_1, x_2, …, x_X, где X – число бит во входной последовательности, *TTI* – число кадров (1, 2, 4 или 8), в которых передается перемежаемая последовательность, причем гарантируется то, что значение X кратно TTI. Выполним следующие действия для осуществления перемежения.

(1) Определим число столбцов перемежителя C из табл. 2 по значению *TTI*. Будем считать, что нумерация строк и столбцов перемежителя начинается с нуля.

(2) Определим число строк перемежителя $R = X / C$.

(3) Запишем входную последовательность в матрицу перемежителя по строкам

$$\begin{pmatrix} x_1 & x_2 & \cdots & x_C \\ x_{C+1} & x_{C+2} & \cdots & x_{2C} \\ \vdots & \vdots & \ddots & \vdots \\ x_{(R-1)C+1} & x_{(R-1)C+2} & \cdots & x_{RC} \end{pmatrix}.$$

(4) Выполним перестановку столбцов матрицы перемежителя в соответствии с табл. 2 в зависимости от значения *TTI*.

(5) Считаем перемеженную последовательность из матрицы перемежителя по столбцам.

Табл. 2.

TTI	C	Значения исходных позиций столбцов
1	1	[0]
2	2	[0, 1]
4	4	[0, 2, 1, 3]
8	8	[0, 4, 2, 6, 1, 5, 3, 7]

Перемежитель №2

В качестве второго перемежителя также используется блоковый перемежитель. Обозначим последовательность, поступающую на перемежитель, u_1, u_2, …, u_U, где U – число бит во входной последовательности. Выполним следующие действия для осуществления перемежения.

(1) Установим число столбцов перемежителя $C = 30$. Будем считать, что нумерация строк и столбцов перемежителя начинается с нуля.

(2) Определим число строк перемежителя $R = \lceil U / C \rceil$.

(3) Запишем входную последовательность в матрицу перемежителя по строкам

$$\begin{pmatrix} y_1 & y_2 & \cdots & y_C \\ y_{C+1} & y_{C+2} & \cdots & y_{2C} \\ \vdots & \vdots & \ddots & \vdots \\ y_{(R-1)C+1} & y_{(R-1)C+2} & \cdots & y_{RC} \end{pmatrix},$$

где $y_k = u_k$ для $k = 1, 2, \ldots, U$ и $y_k = null$ для $k = U + 1, \ldots, RC$. Значения *null* будут отброшены при формировании перемеженной последовательности.

(4) Выполним перестановку столбцов матрицы перемежителя в соответствии с табл. 3.

(5) Считаем перемеженную последовательность из матрицы перемежителя по столбцам, пропуская значения *null*.

Табл. 3.

C	Значения исходных позиций столбцов
30	<0, 20, 10, 5, 15, 25, 3, 13, 23, 8, 18, 28, 1, 11, 21, 6, 16, 26, 4, 14, 24, 19, 9, 29, 12, 2, 7, 22, 27, 17>

Канал синхронизации SCH

Канал синхронизации SCH используется в нисходящем направлении для поиска соты и состоит из двух подканалов: первичного P-SCH и вторичного S-SCH. 10-милисекундные кадры первичного или вторичного канала SCH разделяются на 15 слотов, каждый длиной 2560 чипов. Структура кадра канала SCH показана на рис. 4.

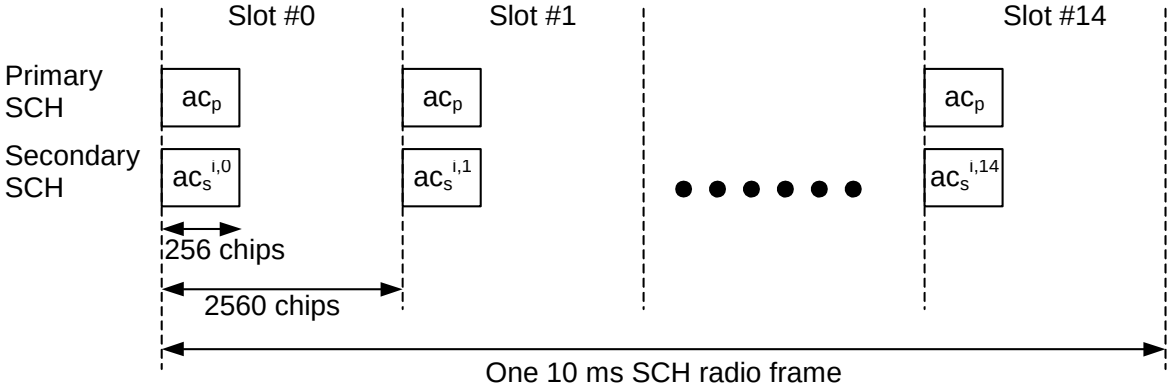


Рис. 4. Структура канала синхронизации SCH

Первичный канал P-SCH состоит из модулирующего кода длины 256 чипов и первичного синхронизирующего кода PSC, обозначенного c_p на рис. 3, передаваемого один раз за слот. Код PSC является одинаковым для каждой соты в системе.

Вторичный канал S-SCH состоит из 15-кратной передаваемой последовательности модулированных кодов, длиной 256 чипов каждый, и вторичного синхронизирующего кода SSC, передаваемого параллельно с первичным каналом P-SCH. Вторичный синхронизирующий код SSC на рис. 3 обозначен как $c_s^{i,k}$, где i = 0, 1, …, 63 – количество групп скремблирующих кодов, а k = 0, 1, …, 14 – номер слота. Каждый код SSC выбирается из набора, содержащего 16 различных кодов длиной 256. Эта последовательность в канале S-SCH показывает, какая из кодовых групп связана со скремблирующим кодом соты в нисходящем направлении.

Общий пилот-канал CPICH

Канал CPICH является нисходящим физическим каналом с фиксированной скоростью, который переносит предопределенную битовую последовательность. На рис. 5 показана структура кадра канала CPICH.

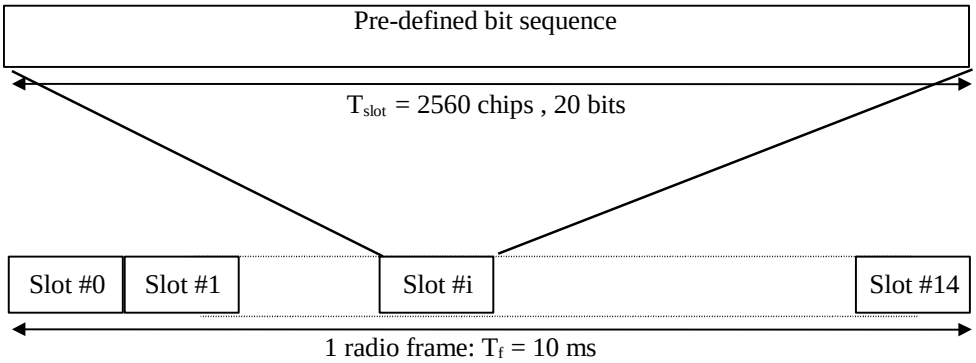


Рис. 5. Структура кадра для общего пилот-канала CPICH

В случае разнесенной передачи (открытого или закрытого цикла), когда в соте используется только нисходящий канал, канала CPICH должен передаваться от обеих антенн с одинаковыми каналообразующими и скремблирующими кодами. В этом случае предопределенная битовая последовательность канала CPICH различна для антенны 1 и

антенны 2 (рис. 6). Для случая отсутствия разнесения используется битовая последовательность антенны 1.

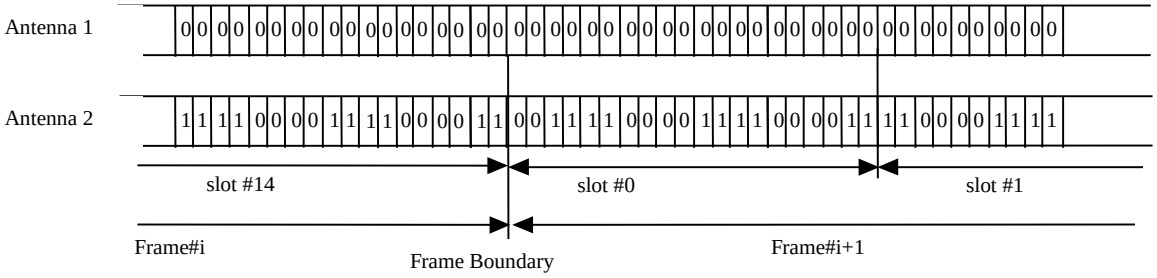


Рис. 6. Модуляционная последовательность для общего пилот-канала

Первичный общий физический канал управления P-CCPCH

Канал P-CCPCH является нисходящим физическим каналом с фиксированной скоростью (30 Кбит/с, SF = 256), используемым для переноса транспортного канала BCH.

На рис. 6 показана структура кадра канала P-CCPCH. Канал P-CCPCH не передается в течение первых 256 чипов каждого слота. Взамен в течение этого периода передаются первичный и вторичный каналы SCH.

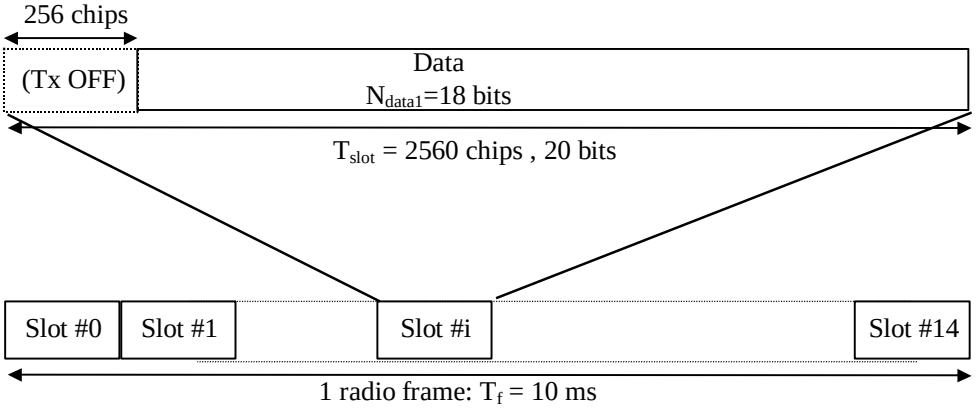


Рис. 7. Структура кадра для канала P-CCPCH