

**Национальный исследовательский университет**  
**«Московский энергетический институт»**  
**Институт радиотехники и электроники**  
**Кафедра радиотехнических систем**

Курсовая работа

По дисциплине: «Аппаратура потребителей спутниковых радионавигационных  
систем»

«Расчет траектории движения спутника GPS по данным с демодулятора его  
сигнала»

ФИО студента: Иванцова Д.Н.

Группа: ЭР-15-17

Вариант №: 17

Дата: \_\_\_\_\_

Подпись: \_\_\_\_\_

ФИО преподавателя: Корогодин И.В.

Оценка: \_\_\_\_\_

МОСКВА, 2022 г

## **Описание работы**

Цель проекта - разработка модулей разбора навигационного сообщения GPS и расчета положения спутника, предназначенных для использования в составе навигационного приемника.

Требования к разрабатываемому программному модулю:

- 1) требования назначения;
- 2) отсутствие утечек памяти;
- 3) малое время выполнения;
- 4) низкий расход памяти;
- 5) корректное выполнение при аномальных входных данных.

Для достижения цели выполняется ряд задач, соответствующих этапам проекта и контрольным мероприятиям:

- 1) разработка модуля разбора символов навигационного сообщения
- 2) расчет положения КА в Matlab/Python и его проверка сторонними сервисами;
- 3) реализация модуля расчета положения КА на C/C++ и его тестирование.

На каждом из этапов действуют следующие правила:

- 1) Взаимодействие осуществляется через github (пул реквесты, комментарии);
- 2) Отчет оформляется по ГОСТ 7.32;
- 3) Этап сдан тогда, когда принят пул реквест.

## Этап 1. Обработка логов навигационного приемника

### 1.1 Описание этапа

Конечная цель всего курсового проекта - получить библиотеку функций на Си++, позволяющую рассчитывать положение спутника GPS по данным с демодулятора его сигнала L1 C/A. На первом этапе реализуем модуль разбора навигационного сообщения до структуры эфемерид, сравним результаты со сторонней программой.

В неизвестной локации установлен навигационный приемник, принимающий сигналы GPS L1C/A и логирующий результаты этого приема в формате NVS BINR. Собранный на пятиминутном интервале файл приложен в архиве под именем BINR.bin, см. таблицу вариантов. Файл содержит наблюдения псевдодальностей и прочих радионавигационных параметров, демодулированные и разобранные данные навигационного сообщения.

Для удобства студентов данные демодулятора продублированы в текстовый файл in.txt. Каждая строка файла содержит данные одного сабфрейма одного навигационного сигнала в формате:

```
1 0 013 0R GpsL1CA # 13 212130404 29 125 53
10001011101010101010101000101001011100001100101001011...
```

где: # 13- номер спутника, 212130404 - счетчик сабфреймов в сигнале, 53- ID сабфрейма в навигационном сообщении, где в первых трех битах содержится номер сабфрейма в фрейме (5 в данном примере), а далее - номер фрейма в сообщении (6 в данном примере), 1000101110... символы с демодулятора в порядке возрастания времени слева направо.

Требуется:

1. Разработать программу, обрабатывающую файл in.txt и выводящую в файл out.txt таблицу эфемерид для спутника согласно варианту в заданном формате;

2. Обработать файл BINR.bin с помощью программы RTKNAVI из состава RTKLIB. Определить день и место проведения наблюдений, значения эфемерид для спутника согласно номеру варианта (меню открывается в левом нижнем углу экрана по нажатию на квадрат);
3. Сравнить полученные таблицы;
4. Оформить код программы и разместить на Github;
5. Оформить отчет по этапу и разместить на Github;
6. Завести Pull Request.

## 1.2 Обработка файла в разработанной программе

Для удобства работы в будущем с помощью Virtual box была поставлена «виртуальная машина» на базе ОС Ubuntu версии 18.04. С той же идеей библиотечные файлы и хидеры сразу же были рассортированы по отдельным папкам и интегрированы в мейн.

Ниже, на рисунке 1, будет представлен образец заданного формата.

```
LNAV Ephemeris (slot = 212130425) =  
  Crs      = -6.537500e+01  
  Dn       = 3.105561e-07      [deg/s]  
  M0       = 129.014897       [deg]  
  Cuc      = -3.531575e-06  
  e        = 1.225188e-02  
  Cus      = 9.512529e-06  
  sqrtA    = 5.153679e+03  
  toe      = 287984  
  Cic      = -1.285225e-07  
  Omega0   = -48.736300      [deg]  
  Cis      = -2.160668e-07  
  i0       = 53.178530      [deg]  
  Crc      = 1.803438e+02  
  omega    = 50.117435      [deg]  
  OmegaDot = -4.758817e-07   [deg/s]  
  iDot     = -3.392870e-08   [deg/s]  
  Tgd      = -1.071021e-08  
  toc      = 287984  
  af2      = 0.000000e+00  
  af1      = 2.614797e-12  
  af0      = -2.331315e-04  
  WN       = 56  
  IODC     = 5  
  URA      = 0  
  Health   = 0  
  IODE2    = 5  
  IODE3    = 5  
  codeL2   = 1  
  L2P      = 0
```

Рисунок 1 — Образец формата данных

По результатам работы программы был получен файл «out.txt» с расшифрованными эфемеридами. На рисунке 2 показано окно текстового редактора с открытым файлом.

```
LNAV Ephemeris (slot = 221472010) =
  Crs = 132.031
  Dn = 2.24691e-07 [deg/s]
  M0 = -120.705 [deg]
  Cuc = 6.99982e-06
  e = 0.0136177
  Cus = 7.82311e-06
  sqrtA = 5153.76
  toe = 93600
  Cic = 1.73226e-07
  Omega0 = -166.61 [deg]
  Cis = 1.91852e-07
  i0 = 56.1608 [deg]
  Crc = 237.844
  omega = -85.8694 [deg]
  OmegaDot = -4.34832e-07 [deg/s]
  iDot = 1.14801e-08 [deg/s]
  Tgd = 0
  toc = 93600
  af2 = 0
  af1 = 5.00222e-12
  af0 = 0.000574787
  WN = 149
  IODC = 71
  URA = 0
  Health = 0
  IODE2 = 71
  IODE3 = 71
  codeL2 = 1
  L2P = 1
```

Текст    Ширина табуляции: 8    Стр 21, Стлб 12    ВСТ

Рисунок 2 — Файл «out.txt»

### Этап 1.3 Обработка с помощью RTKNAVI

Берём с гита одну из версий программного пакета RTKLIV. Запускаем под нужной системой поддиректорию RTKNAVI. Нажимаем на кнопку I в верхнем правом углу программы(в данном случае была использована ОС Windows), выставляем параметры, как показано на рисунке 3, остальное — не трогаем (по умолчанию).

Input Streams

Input Stream	Type	Opt	Cmd	Format	Opt
<input checked="" type="checkbox"/> (1) Rover	File	...	...	NVS BINR	...
<input type="checkbox"/> (2) Base Station	Serial	...	...	RTCM 2	...
<input type="checkbox"/> (3) Correction	Serial	...	...	RTCM 2	...

Transmit NMEA GGA to Base Station

OFF 0.000000000 0.000000000 0.000

Reset Cmd Max Baseline 10 km

Input File Paths

C:\BINR.bin

☐ Time x1 + 0 s ☐ 64bit

OK Cancel

Рисунок 3 — Настройки RTKNAVI

Нажимаем start и видим картину, как на рисунке 4, где нам высветились ОСШ спутников, а так же координаты.

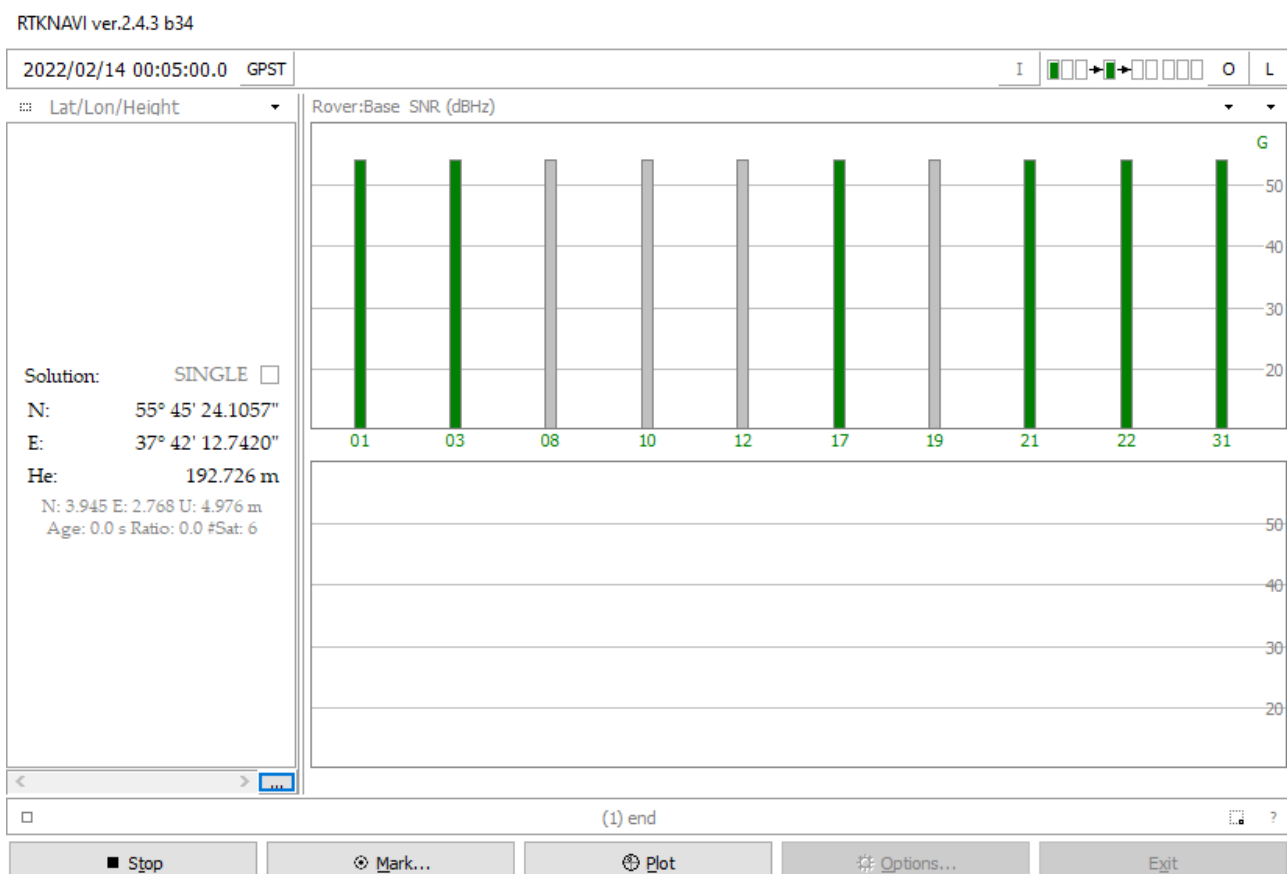


Рисунок 4 — Интерфейс RTKNAVI

Переходим в расчёт эфемеридов, нажав в нижнем левом углу на квадрат и выбрав NAV DATA. Выбираем GPS и ONLY OK(наш спутник-то точно «окей»), чтобы упростить поиск. На рисунках 5 и 6 показан «ответ» программы:

RTKNAVI ver.2.4.3 b34: RTK Monitor

Nav Data	GPS	Only OK	Current 1											Close
SAT	PRN	Statu	IDOE	IDOC	URA	SVH	Toe	Toc	Ttrans	A (m)	e	i0 (°)	Ω0 (°)	ω (°)
G01	1	OK	5911	23	0	000	2022/02/14 00:00:00	2022/02/14 00:00:00	2022/05/23 05:58:46	26560304.325	0.01137974	56.53057	-109.43551	50.51432
G03	3	OK	2827	11	0	000	2022/02/14 01:59:44	2022/02/14 01:59:44	2022/05/23 05:58:46	26560504.265	0.00389213	55.72485	-50.10616	54.89203
G08	8	OK	1028	4	0	000	2022/02/14 01:59:44	2022/02/14 01:59:44	2022/05/23 05:58:46	26560852.794	0.00728541	55.29557	-171.29825	5.90629
G10	10	OK	1182	46	0	000	2022/02/14 02:00:00	2022/02/14 02:00:00	2022/05/23 05:58:46	26560464.021	0.00750119	55.71486	-50.26591	-144.08942
G12	12	OK	1773	69	0	000	2022/02/14 02:00:00	2022/02/14 02:00:00	2022/05/23 05:58:46	26559621.018	0.00868590	55.59347	133.97619	71.43227
G17	17	OK	1824	71	0	000	2022/02/14 02:00:00	2022/02/14 02:00:00	2022/05/23 05:58:46	26561289.658	0.01361771	56.16076	-166.60982	-85.86944

Рисунок 5 — Эфемериды для заданных спутников часть 1

RTKNAVI ver.2.4.3 b34: RTK Monitor

Nav DataGPSOnly OKCurrent 1

Close

ω (°)	M0 (°)	Δn (°/s)	Ωdot (°/s)	IDOT (°/s)	Af0 (ns)	Af1 (ns/s)	Af2 (ns/s)	TGD1 (ns)	TGD2 (ns)	Cuc (rad)	Cus (rad)	Crc (m)	Crz (m)	Cic (rad)	Cis (rad)	Code	Flag
50.51432	54.55674	2.4630E-07	-4.8124E-07	-6.7735E-09	432151.82	-0.0095	0.0000	5.12	0.00	7.7300E-07	1.1530E-06	371.969	11.719	-1.1735E-07	1.1362E-07	001	00
54.89203	48.47481	2.3572E-07	-4.5681E-07	-1.7149E-08	-120342.71	-0.0172	0.0000	1.86	0.00	-6.6254E-06	7.0781E-06	248.250	-127.344	-4.4703E-08	3.7253E-08	001	00
5.90629	-115.58396	2.4679E-07	-4.5104E-07	9.9453E-09	-56090.30	-0.0015	0.0000	5.12	0.00	5.2750E-06	8.3819E-06	223.000	105.906	-5.5879E-09	-1.3225E-07	001	00
-144.08942	-1.69731	2.3953E-07	-4.6279E-07	-1.7312E-08	-319812.92	-0.0105	0.0000	2.33	0.00	-5.8059E-06	6.2771E-06	262.969	-110.031	2.2352E-08	6.8918E-08	001	00
71.43227	38.93650	2.4376E-07	-4.5196E-07	-3.1125E-08	-170663.46	-0.0057	0.0000	-12.57	0.00	3.0547E-07	9.8143E-06	195.250	8.406	-1.3411E-07	-9.3132E-09	001	00
-85.86944	-120.70525	2.2469E-07	-4.3483E-07	1.1480E-08	574786.66	0.0050	0.0000	-11.18	0.00	6.9998E-06	7.8231E-06	237.844	132.031	1.7323E-07	1.9185E-07	001	00

Рисунок 6 — Эфемериды для заданных спутников часть 2



#### **Этап 1.4 Сравнение полученных значений**

Сравнив результаты из этапов 1.2 и 1.3, можно сделать вывод, что разработанная программа получает правильные значения, так как данные совпадают, что и требовалось доказать.

## Приложение 1

Код для первого этапа

CMakeLists.txt:

```
1 cmake_minimum_required(VERSION 2.8)
2
3 project(etap1_code)
4
5 add_executable(${PROJECT_NAME} src/main.cpp
6   ... header/head.h
7   ... src/otpravit.cpp
8   ... src/data.cpp
9   ... src/toint.cpp
10  ... src/sohranit.cpp
11  ...)
```

main.cpp:

```
1 #include "../header/head.h"
2
3 using namespace std;
4
5 int main(void)
6 {
7   ... frame.pack;
8   ... otpravit(&pack);
9   ... Ephemeris *Eph = (Ephemeris*) malloc(sizeof(Ephemeris));
10  ... razobrat(Eph, &pack);
11  ... sohranit(Eph);
12  ... free(Eph);
13 }
```

head.h:

```
1  #ifndef HEAD_H
2  #define HEAD_H
3
4  #include <iostream>
5  #include <fstream>
6  #include <stdlib.h>
7  #include <cmath>
8  #include <stdio.h>
9
10 #define Cnst_2E5 pow(2,-5)
11 #define Cnst_2E55 pow(2,-55)
12 #define Cnst_2E43 pow(2,-43)
13 #define Cnst_2E31 pow(2,-31)
14 #define Cnst_2E4 pow(2,4)
15 #define Cnst_2E19 pow(2,-19)
16 #define Cnst_2E29 pow(2,-29)
17 #define Cnst_2E33 pow(2,-33)
18 #define SC 180
19
20 struct Ephemeris {
21     ...float Crs;
22     ...float Dn;
23     ...float M0;
24     ...float Cuc;
25     ...float e;
26     ...float Cus;
27     ...float sqrtA;
28     ...uint32_t toe;
29     ...float Cic;
30     ...float Omega0;
31     ...float Cis;
32     ...float i0;
33     ...float Crc;
34     ...float omega;
35     ...float OmegaDot;
36     ...float iDot;
37     ...int16_t Tgd;
```

```

37     ....uint16_t·Tgd;
38     ....uint32_t·toc;
39     ....float·af2;
40     ....float·af1;
41     ....float·af0;
42     ....uint16_t·WN;
43     ....uint16_t·IODC;
44     ....uint32_t·URA;
45     ....uint32_t·Health;
46     ....uint16_t·IODE2;
47     ....uint16_t·IODE3;
48     ....bool·codeL2;
49     ....bool·L2P;
50     ....uint32_t·pack;
51 };
52
53 struct·frame·{
54     ....uint32_t·pack;
55     ....std::string·Cnst1;
56     ....std::string·Cnst2;
57     ....std::string·Cnst3;
58 };
59
60 void·otpravit(frame·*Rpack);
61 uint32_t·gruppировka(std::string·Cnst,·uint16_t·start1,·int·dlit1,·uint16_t·start2,·int·dlit2);
62 void·sohranit(Ephemeris·*·Eph);
63 void·razobrat(Ephemeris·*·Eph,·frame·*pack);
64 int32_t·data(std::string·Cnst,·int32_t·start,·int·dlit);
65
66 #endif·//·HEAD_H

```

data.cpp:

```

#include·"../header/head.h"

int32_t·data(std::string·Cnst,·int32_t·start,·int·dlit)·{
    ....int32_t·retrn·=·0;
    ....int32_t·Rretrn·=·0;
    ....for·(int·i·=·start;·i·<·start+dlit;·i++)·{
        ....retrn·=·(retrn·|·((Cnst[i-1]·==·'1')·?·1·:·0));
        ....if·(i·<·start+dlit-1){
            ....retrn·=·retrn<<1;
        }
    }
    ....return·retrn;
}

```

otpravit.cpp:

```
1 | #include "../header/head.h"
2 |
3 | ▼ void otpravit(frame *Rpack)
4 | {
5 |     std::string path = "in.txt";
6 |     std::ifstream fin;
7 |     fin.open(path);
8 |     ▼ while (!fin.eof()) {
9 |         int N = 3;
10 |         std::string empty;
11 |         int nmb_r_stlt;
12 |         uint32_t pack;
13 |         uint32_t frameFrameNum;
14 |         std::string str;
15 |         uint32_t pack_Cnst1;
16 |         uint32_t pack_Cnst2;
17 |         uint32_t pack_Cnst3;
18 |         int mass[N];
19 |         ▼ for (int i=0; i<N; ++i)
20 |             {
21 |                 fin >> mass[i];
22 |             }
23 |         fin >> empty >> empty >> empty; //
24 |         fin >> nmb_r_stlt >> pack >> empty >> empty >> frameFrameNum;
25 |
26 |         fin >> str;
27 |         ▼ if (nmb_r_stlt == 17 and pack >= 604800/6) {
28 |             ▼ if (frameFrameNum == 1)
29 |                 {
30 |                     pack_Cnst1 = pack;
31 |                     Rpack->Cnst1 = str;
32 |                 }
33 |             ▼ else if (frameFrameNum == 2)
34 |                 {
35 |                     pack_Cnst2 = pack;
36 |                     Rpack->Cnst2 = str;
37 |                 }
38 |             ▼ else if (frameFrameNum == 3)
39 |                 {
40 |                     pack_Cnst3 = pack;
41 |                     Rpack->Cnst3 = str;
42 |                 }
43 |             ▼ if (pack_Cnst1 + 1 == pack_Cnst2 and pack_Cnst2 + 1 == pack_Cnst3) {
44 |                 Rpack->pack = pack_Cnst1;
45 |                 return;
```

sohranit.cpp:

```
1 | #include "../header/head.h"
2 |
3 | void sohranit(Ephemeris* Eph)
4 | {
5 |     std::ofstream fout;
6 |     std::string path = "out.txt";
7 |     fout.open(path);
8 |     fout << std::endl << "LNAV Ephemeris (slot = " << Eph->pack << ") = " << std::endl;
9 |     fout << "\t Crs = " << Eph->Crs << std::endl;
10 |    fout << "\t Dn = " << Eph->Dn << "\t\t[deg/s]" << std::endl;
11 |    fout << "\t M0 = " << Eph->M0 << "\t\t\t[deg]" << std::endl;
12 |    fout << "\t Cuc = " << Eph->Cuc << std::endl;
13 |    fout << "\t e = " << Eph->e << std::endl;
14 |    fout << "\t Cus = " << Eph->Cus << std::endl;
15 |    fout << "\t sqrtA = " << Eph->sqrtA << std::endl;
16 |    fout << "\t toe = " << Eph->toe << std::endl;
17 |    fout << "\t Cic = " << Eph->Cic << std::endl;
18 |    fout << "\t Omega0 = " << Eph->Omega0 << "\t\t[deg]" << std::endl;
19 |    fout << "\t Cis = " << Eph->Cis << std::endl;
20 |    fout << "\t i0 = " << Eph->i0 << "\t\t\t[deg]" << std::endl;
21 |    fout << "\t Crc = " << Eph->Crc << std::endl;
22 |    fout << "\t omega = " << Eph->omega << "\t\t[deg]" << std::endl;
23 |    fout << "\t OmegaDot = " << Eph->OmegaDot << "\t[deg/s]" << std::endl;
24 |    fout << "\t iDot = " << Eph->iDot << "\t\t\t[deg/s]" << std::endl;
25 |    fout << "\t Tgd = " << Eph->Tgd << std::endl;
26 |    fout << "\t toc = " << Eph->toc << std::endl;
27 |    fout << "\t af2 = " << Eph->af2 << std::endl;
28 |    fout << "\t af1 = " << Eph->af1 << std::endl;
29 |    fout << "\t af0 = " << Eph->af0 << std::endl;
30 |    fout << "\t WN = " << Eph->WN << std::endl;
31 |    fout << "\t IODC = " << Eph->IODC << std::endl;
32 |    fout << "\t URA = " << Eph->URA << std::endl;
33 |    fout << "\t Health = " << Eph->Health << std::endl;
34 |    fout << "\t IODE2 = " << Eph->IODE2 << std::endl;
35 |    fout << "\t IODE3 = " << Eph->IODE3 << std::endl;
36 |    fout << "\t codeL2 = " << Eph->codeL2 << std::endl;
37 |    fout << "\t L2P = " << Eph->L2P << std::endl;
38 |    fout.close();
39 | }
```

toint.cpp:

```
1 | #include "../header/head.h"
2 |
3 | ▼ int32_t compl2int(uint32_t retn, int dlit_frame){
4 |
5 |     ... int32_t Rretn = 0;
6 |     ▼ ... if (dlit_frame == 8){
7 |     ▼ ...     if (bool((1<7) & retn)){
8 |         ...     retn |= 0xFFFFF00;
9 |         ...     Rretn = ~(retn-1);
10 |        ...     return --Rretn;
11 |        ...     }
12 |    ... }
13 | ▼ ... if (dlit_frame == 14){
14 | ▼ ...     if (bool((1<13) & retn)){
15 |         ...     retn |= 0xFFFFC000;
16 |         ...     Rretn = ~(retn-1);
17 |         ...     return --Rretn;
18 |         ...     }
19 |    ... }
20 | ▼ ... if (dlit_frame == 16){
21 | ▼ ...     if (bool((1<15) & retn)){
22 |         ...     retn |= 0xFFFF0000;
23 |         ...     Rretn = ~(retn-1);
24 |         ...     return --Rretn;
25 |         ...     }
26 |    ... }
27 | ▼ ... if (dlit_frame == 22){
28 | ▼ ...     if (bool((1<21) & retn)){
29 |         ...     retn |= 0xFFC00000;
30 |         ...     Rretn = ~(retn-1);
31 |         ...     return --Rretn;
32 |         ...     }
33 |    ... }
34 | ▼ ... if (dlit_frame == 24){
35 | ▼ ...     if (bool((1<23) & retn)){
36 |         ...     retn |= 0xFF000000;
37 |         ...     Rretn = ~(retn-1);
38 |         ...     return --Rretn;
39 |         ...     }
40 |    ... }
41 | ▼ ... if (dlit_frame == 32){
42 | ▼ ...     if (bool((1<31) & retn)){
43 |         ...     retn |= 0x00000000;
44 |         ...     Rretn = ~(retn-1);
45 |         ...     return --Rretn;

```

```

51  uint32_t gruppirovka(std::string Cnst, uint16_t start1, int dlit1, uint16_t start2, int dlit2) {
52      uint32_t retn = 0;
53      for (int i = start1; i < start1 + dlit1; i++) {
54          retn = (retn | ((Cnst[i-1] == '1') ? 1 : 0)) << 1;
55      }
56      for (int i = start2; i < start2 + dlit2; i++) {
57          retn = retn | ((Cnst[i-1] == '1') ? 1 : 0);
58          if (i < start2 + dlit2 - 1) {
59              retn = retn << 1;
60          }
61      }
62      return retn;
63  }

65  void razobrat(Ephemeris* Eph, frame* pack) {
66      Eph->pack = pack->pack;
67      Eph->CrS = compl2int(data(pack->Cnst2, 69, 16), 16) * Cnst_2E5;
68      Eph->Dn = compl2int(data(pack->Cnst2, 91, 16), 16) * Cnst_2E43 * SC;
69      Eph->M0 = compl2int(gruppirovka(pack->Cnst2, 107, 8, 121, 24), 32) * Cnst_2E31 * SC;
70      Eph->Cuc = compl2int(data(pack->Cnst2, 151, 16), 16) * Cnst_2E29;
71      Eph->e = gruppirovka(pack->Cnst2, 167, 8, 181, 24) * Cnst_2E33;
72      Eph->Cus = compl2int(data(pack->Cnst2, 211, 16), 16) * Cnst_2E29;
73      Eph->sqrtA = gruppirovka(pack->Cnst2, 227, 8, 241, 24) * Cnst_2E19;
74      Eph->toe = data(pack->Cnst2, 271, 16) * pow(2, 4);
75      Eph->Cic = compl2int(data(pack->Cnst3, 61, 16), 16) * Cnst_2E29;
76      Eph->Omega0 = compl2int(gruppirovka(pack->Cnst3, 77, 8, 91, 24), 32) * Cnst_2E31 * SC;
77      Eph->Cis = compl2int(data(pack->Cnst3, 121, 16), 16) * Cnst_2E29;
78      Eph->i0 = compl2int(gruppirovka(pack->Cnst3, 137, 8, 151, 24), 32) * Cnst_2E31 * SC;
79      Eph->Crc = compl2int(data(pack->Cnst3, 181, 16), 16) * Cnst_2E5;
80      Eph->omega = compl2int(gruppirovka(pack->Cnst3, 197, 8, 211, 24), 32) * Cnst_2E31 * SC;
81      Eph->OmegaDot = compl2int(data(pack->Cnst3, 241, 24), 24) * Cnst_2E43 * SC;
82      Eph->iDot = compl2int(data(pack->Cnst3, 279, 14), 14) * Cnst_2E43 * SC;
83      Eph->Tgd = compl2int(data(pack->Cnst1, 197, 8), 8) * Cnst_2E31;
84      Eph->toc = compl2int(data(pack->Cnst1, 219, 16), 16) * Cnst_2E4;
85      Eph->af2 = compl2int(data(pack->Cnst1, 241, 8), 8) * Cnst_2E55;
86      Eph->af1 = compl2int(data(pack->Cnst1, 249, 16), 16) * Cnst_2E43;
87      Eph->af0 = compl2int(data(pack->Cnst1, 271, 22), 22) * Cnst_2E31;
88      Eph->WN = data(pack->Cnst1, 61, 10);
89      Eph->IODC = gruppirovka(pack->Cnst1, 83, 2, 211, 8);
90      Eph->URA = data(pack->Cnst1, 73, 4);
91      Eph->Health = Eph->IODE2 = data(pack->Cnst1, 73, 6);
92      Eph->IODE2 = data(pack->Cnst2, 61, 8);
93      Eph->IODE3 = data(pack->Cnst3, 271, 8);
94      Eph->codeL2 = data(pack->Cnst1, 71, 2);
95      Eph->L2P = pack->Cnst1[90];
96  }

```