УТВЕРЖДЕН

РАЯЖ.00168-01 13 01-ЛУ

МИКРОСХЕМА ИНТЕГРАЛЬНАЯ 1892ВМ10Я
БИБЛИОТЕКА НАВИГАЦИОННОГО ПО

Описание программы

РАЯЖ.00168-01 13 01

CD-R

Листов 24

2012

Литера

Инв. № подл.	Подп. и дата	Взам.инв.№	Инв.№ дубл.	Подп. и дата

АННОТАЦИЯ

В документе «Микросхема интегральная 1892ВМ10Я. Библиотека навигационного ПО. Описание программы» РАЯЖ.00168-01 13 01 приводится описание библиотеки NVLib, ее интерфейсы и способы работы с ней.

NVLib представляет собой библиотеку программного обеспечения, предназначенную для определения точного времени и координат по сигналам спутниковых систем GPS и GLONASS. Библиотека представляет собой программный модуль, имеющий интерфейс в виде набора функций, предназначенный для встраивания в прикладные программы и программные комплексы, реализованные на микросхеме 1892ВМ10Я.

СОДЕРЖАНИЕ

1. ОБЩИЕ СВЕДЕНИЯ

1.1. Обозначение и наименование программы

РАЯЖ.00168-01

Библиотека навигационного ПО (далее по тексту – NVLib).

1.2. Программное обеспечение и используемые технические средства

1.2.1. Для использования NVLib на ПЭВМ должна быть установлена ОС Windows или Linux, а также среда разработки программного обеспечения для процессоров NVCom – MultiCore Studio 3M (РАЯЖ.00167-01).

1.2.2. Для установки и функционирования среды разработки uOS рекомендуется ПЭВМ со следующими характеристиками:

- процессор 2000 МГц Intel Pentium 4;
- оперативная память не менее 512 Мбайт;
- магнитный жесткий диск не менее 2,5 Гбайт.

Для загрузки образа целевой системы в постоянную память модуля, а также для пошаговой отладки программ требуется эмулятор интерфейсов USB-JTAG ЛЦКБ.467133.002.

1.3. Языки программирования

1.3.1. Библиотека навигационного ПО написана на языках С и ассемблера для архитектуры MIPS-32.

2. ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ

2.1. Назначение программы

2.1.1. Библиотека NVLib может применяться в промышленных и коммерческих электронных изделиях возимого или носимого типа (коммуникационных системах, измерительном оборудовании, системах мониторинга и т.п.), построенных с использованием микросхемы 1892ВМ10Я, включающей, в частности, аппаратный блок корреляторов и ядро микроконтроллера архитектуры MIPS-32.

2.1.2. В задачи библиотеки входят:

- управление многоканальным коррелятором микросхемы интегральной 1892ВМ10Я;
- поиск и слежение за сигналами спутников GPS/ГЛОНАСС;
- прием и декодирование информационных сообщений навигационных спутников GPS/ГЛОНАСС;
- решение навигационной задачи (расчет времени и местоположения);
- обслуживание интерфейса диагностики и отладки.

3. ОПИСАНИЕ ЛОГИЧЕСКОЙ СТРУКТУРЫ

3.1. Модульный принцип

3.1.1. Система построена по модульному принципу. Основным модулем системы является модуль NVCore, написанный на языке C и являющийся платформенно независимым. Остальные модули либо являются опциональными, либо могут изменяться в зависимости от типа платформы и приложения.

3.2. Состав библиотеки и ее место в комплексе ПО

3.2.1. Структура библиотеки NVLib приведена на рис. 1. Модули, составляющие части библиотеки обведены двойными рамками. Каждый модуль можно условно отнести к одному из трех множеств:

1) независимые от платформы и приложения;

2) зависимые от приложения;

3) зависимые от платформы.

3.2.2. В библиотеку входят следующие основные модули.

1) HAL (RTOS/HW) – Hardware Adaptation Level – модуль, обеспечивающий инициализацию и диспетчеризацию процессов библиотеки NVlib, а также платформенно независимый программный интерфейс к аппаратным ресурсам платформы, за исключением коррелятора (последний вынесен в отдельный модуль CHAL).

HAL не является постоянной частью библиотеки. Этот модуль должен разрабатываться для конкретной конфигурации аппаратуры и ПО разработчиком приложения или платформы. HAL может быть построен, в том числе, с применением RTOS.

Для инициализации и диспетчеризации процессов, библиотека NVlib поставляет интерфейс NVL-HAL (e1), содержащий функции инициализации и вызова нескольких процессов с разным темпом и уровнем приоритетности.

Для абстракции аппаратных ресурсов HAL должен поставлять интерфейс HAL-NVL (e2), содержащий следующие группы функций:

- чтение/запись/стирание FLASH-памяти;
- чтение/запись BACKUP-памяти (BRAM);

- возврат/установка реального времени (доступ к RTC или другому механизму);

Структура библиотеки NVLib

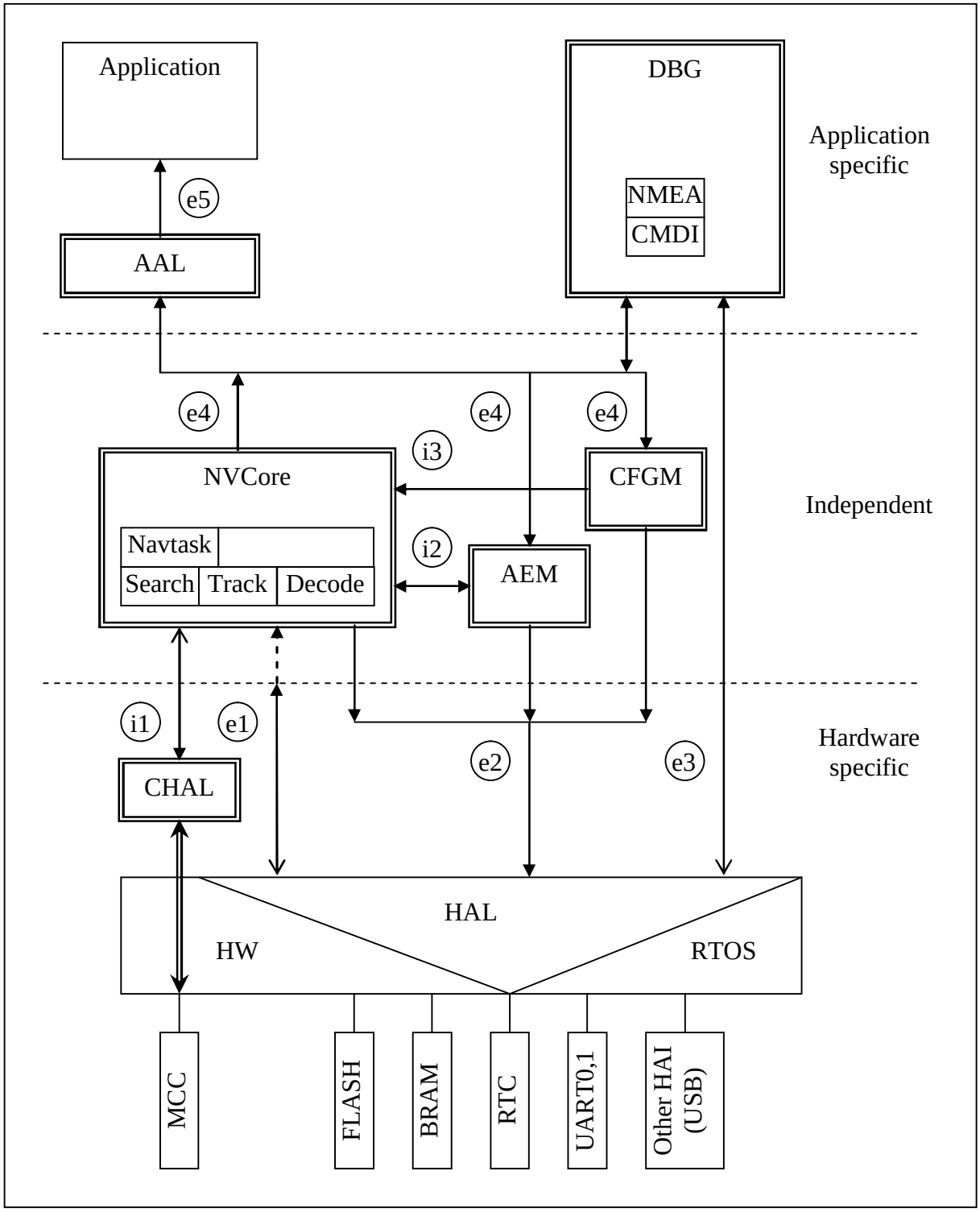

Рисунок 1

2) CHAL – Correlation Hardware Adaptation Level – модуль адаптации блока аппаратных корреляторов. Работает непосредственно с аппаратурой коррелятора и обеспечивает независимость других модулей от него. Поставляемый интерфейс: CHAL-NVC (i1);

3) NVCore – ядро навигационной библиотеки, выполняющее задачи поиска и слежения за навигационными сигналами, декодирования навигационных сообщений и решения навигационных задач. Поставляемые интерфейсы:

- NVL-AEM (i2) – чтение/запись актуальных эфемерид и альманахов;
- NVL-CFG (i3) – установка/чтение конфигурации ядра;
- NVL-APP (e4) –интерфейс приложений в части передачи навигационных данных;

4) AEM – Almanac and Ephemeris Management – модуль управления альманахами и эфемеридами. Обеспечивает хранение, синхронизацию альманахов и эфемерид в оперативной памяти, FLASH и BRAM и доступ к ним со стороны других модулей. Поставляемый интерфейс: NVL-APP (e4) –интерфейс приложений в части управления эфемеридами и альманахами;

5) CFGM – Configuration Management – модуль управления конфигурацией. Обеспечивает хранение, синхронизацию конфигурации в оперативной памяти, FLASH и BRAM и доступ к ним со стороны других модулей. Поставляемый интерфейс: NVL-APP (e4) – интерфейс приложений в части управления конфигурацией;

6) AAL – Application Adaptation Level – модуль адаптации к простому приложению. Обеспечивает передачу минимального рекомендуемого объема навигационных данных для приложения. Разрабатывается специально для конкретного приложения и осуществляет взаимодействие между навигационной библиотекой и приложением посредством интерфейса NVL-APP. Интерфейс (e5) разрабатывается с учетом требований приложения и не является частью библиотеки;

7) DBG – отладочный модуль. Обеспечивает формирование потока навигационных сообщений в формате NMEA и прием текстовых команд в формате NMEA+. Может быть использован в «stand-alone» приемнике и других приложениях, где используются сообщения в формате NMEA. Поставляемый интерфейс: DBG-HAL (e3). DBG содержит следующие подмодули, которые могут быть использованы независимо:

- NMEA – формирование NMEA сообщений из двоичных данных;
- CMDI – декодирование текстовых команд в формате NMEA+.

3.3. Поддерживаемые архитектуры

3.3.1. Аппаратно независимая часть

3.3.1.1. Аппаратно независимая часть может исполняться на процессорах любой архитектуры, удовлетворяющих следующим минимальным требованиям:

- разрядность архитектуры 32 бита;
- вычислительная мощность 40-50 MIPS;
- наличие сопроцессора для вычислений с плавающей точкой.

3.3.2. Аппаратно зависимая часть

3.3.2.1. Аппаратно зависимые модули библиотеки в настоящее время могут исполняться только на навигационных процессорах серии Навиком.

4. ИНТЕРФЕЙСЫ БИБЛИОТЕКИ

4.1. Термины и соглашения, использованные при описании интерфейсов

4.1.1. Для удобства описания интерфейсов использованы следующие понятия:

1) модуль, поставляющий сервис в рамках определенного интерфейса, является сервером этого интерфейса. Соответственно, модуль, использующий сервис в рамках определенного интерфейса, является клиентом этого интерфейса. Интерфейс определяется со стороны сервера. В общем случае интерфейсы в NVLib определены таким образом, что только один модуль является сервером. В то же время модуль может быть сервером для одних интерфейсов и клиентом для других. Для удобства наряду с термином «сервер» или «поставщик сервиса» в ряде случаев будет использоваться термин «поставщик интерфейса»;

2) сервис может предоставляться одним из двух способов:

- прямой вызов функций сервера клиентом. В этом случае функция выполняется в процессе клиента;
- обратный вызов функций клиента сервером. В этом случае клиент сначала регистрирует свою функцию (callback function) в сервере вызовом соответствующей функции установки функции обратного вызова. Далее по необходимости сервер вызывает функцию обратного вызова клиента, которая выполняется в процессе сервера;

3) функции разделены на две категории:

- функции-запросы, время выполнения которых значительно меньше (в 100 и более раз), чем период эпохи (1 мс);
- функции-обработчики, время выполнения которых сравнимо или больше, чем период эпохи (1 мс).

4.2. Интерфейс NVL-HAL

4.2.1. Общие сведения

4.2.1.1. Интерфейс NVL-HAL предназначен для инициализации библиотеки и управления выполнением основных процессов NVLib. Поставщиком интерфейса является NVCore. HAL должен обеспечить вызов функций интерфейса в правильном порядке с указанными временными параметрами.

Выполнение функций (задач) GNSS_epoch, GNSS_DSP_task и GNSS_DSP_done привязано к сигналу прерывания эпохи. По сигналу прерывания эпохи эти задачи должны выполниться друг за другом и завершить свою работу до возникновения следующего сигнала эпохи.

4.2.2. Функции интерфейса NVL-HAL

4.2.2.1. Основные функции-обработчики навигационной библиотеки.

void GNSS_init();

Назначение: функция инициализации навигационной библиотеки.

Ядро: RISC.

Период вызова: однократно из HAL.

Прерываемость: нет.

void GNSS_epoch();

Назначение: точка входа в ассемблерный обработчик эпохи на RISC.

Ядро: RISC.

Период вызова: 1 мс; функция должна вызываться из HAL по прерыванию эпохи (от коррелятора).

Прерываемость: нет. Функция реального времени.

Время выполнения (максимальное): не более 45 мкс на частоте RISC 200 МГц.

Допустимая латентность вызова от прерывания по эпохе: 100 мкс.

Во избежание потерь процессорного времени желательно расположение кода функции во внутренней памяти – CRAM или XY_MEM.

void GNSS_track();

Назначение: функция обработчика поиска и слежения.

Ядро: RISC.

Период вызова: ~10 мс; функция должна вызываться из HAL по таймеру или по запросу из функции обратного вызова GNSS_start_track_callback_f. Возможна настройка на другой период вызова.

Прерываемость: да. Функция реального времени (пропуски вызова критичны для устойчивости работы приемника, функция должна завершиться к моменту следующего вызова).

Время выполнения (максимальное): TBD мкс на частоте RISC 200 МГц. Функция может выполняться в течение нескольких эпох.

Допустимая латентность вызова от события таймера или вызова функции GNSS_start_track_callback_f: 5 мс.

Если установлена функция обратного вызова GNSS_start_solve_callback, то она периодически инициирует вызов функции расчета координат **`GNSS_solve()`**.

void GNSS_solve();

Назначение: функция решения навигационной задачи (расчет координат на основе текущей информации о координатах спутников и псевдодальностях).

Ядро: RISC.

Период вызова: 1 с. Функция должна вызываться из HAL по таймеру или по запросу из функции обратного вызова GNSS_start_solve_callback_f.

Прерываемость: да. Фоновая задача (пропуски вызова не критичны для устойчивости работы приемника).

Время выполнения (среднее/максимальное) 0.1/0.5 с на частоте RISC 200 МГц. Функция может выполняться в течение нескольких эпох.

Допустимая латентность вызова от таймера или вызова функции GNSS_start_solve_callback: 100 мс.

4.2.3. Функции обратного вызова

4.2.3.1. Функции обратного вызова предназначены для реализации механизма, когда периодичность вызова функций реального времени определяется собственным

механизмом NVLib. Использование этого механизма является необязательным, но его применение позволяет упростить реализацию HAL при использовании RTOS.

При установке функции обратного вызова GNSS_track_callback (GNSS_solve_callback) с помощью функции GNSS_set_track_callback (GNSS_set_solve_callback), NVLib будет сигнализировать HAL о необходимости вызова функций GNSS_track (GNSS_solve). Если функции обратного вызова не установлены, то функции GNSS_track и GNSS_solve должны вызываться со стороны HAL по фиксированной циклограмме (таймеру). Функции выполняются на RISC-ядре.

void GNSS_set_track_callback(GNSS_start_track_callback_f func);

Назначение: установка указателя на функцию обратного вызова для передачи запроса на вызов задачи GNSS_track() со стороны HAL.

Период вызова: функция должна вызываться со стороны HAL однократно при инициализации.

Прерываемость: да.

Время выполнения: не критично.

typedef void (GNSS_start_track_callback_f *)(void);

Назначение: запрос на вызов задачи GNSS_track(). Функция должна определяться в HAL и передаваться в NVLib с помощью функции GNSS_set_track_callback.

После вызова данной функции, HAL должен запустить функцию GNSS_track() не позднее, чем через 5 мс.

Период вызова: 10 - 50 мс, вызывается со стороны NVLib.

Прерываемость: нет.

Время выполнения: более 10 мкс.

void GNSS_set_solve_callback(GNSS_start_solve_callback_f func);

Назначание: установка указателя на функцию обратного вызова для передачи запроса на вызов задачи GNSS_solve() со стороны HAL.

Период вызова: должна вызываться со стороны HAL однократно при инициализации.

Прерываемость: да.

Время выполнения: не критично.

typedef void (GNSS_start_solve_callback_f *)(void);

Назначание: запрос на вызов задачи GNSS_solve(). Должна определяться в HAL и передаваться в NVLib с помощью функции GNSS_set_solve_callback.

После вызова данной функции, HAL должен запустить функцию GNSS_solve() не позднее, чем через 100 мс.

Период вызова: 0.2 с или больше, в соответствии с темпом решения навигационной задачи; вызывается со стороны NVLib.

Прерываемость: нет.

Время выполнения: не более 10 мкс.

4.2.4. Устаревшие функции

4.2.4.1. Данные функции имеют тривиальную реализацию (возврат из функции непосредственно после вызова) в NVLib версии 2.x и оставлены для совместимости с платформами, работающими с NVLib версией 1.x. В новых реализациях HAL эти функции использоваться не должны.

typedef void (GNSS_DSP_run_callback_f*)(void); GNSS_set_DSP_run_callback (GNSS_DSP_run_callback_f func);

Функция не используется в версии библиотеки 2.0 и старше, и оставлена для совместимости.

void GNSS_DSP_task(void);

Функция не используется в версии библиотеки 2.0 и старше, и оставлена для совместимости.

void GNSS_DSP_done();

Функция не используется в версии библиотеки 2.0 и старше, и оставлена для совместимости.

4.3. Интерфейс HAL-NVL

4.3.1. Общие сведения

4.3.1.1. Интерфейс HAL-NVL является абстракцией аппаратуры платформы, за исключением коррелятора, для библиотеки NVLib. Поставщиком интерфейса является HAL – модуль, не входящий в NVLib и разрабатываемый для портирования NVLib на конкретную аппаратно-программную платформу. Интерфейс содержит несколько функциональных групп.

4.3.2. Функции интерфейса NVL-HAL

4.3.2.1. Доступ к энергонезависимой памяти:

const int HAL_bram_seg_size;

Размер сегмента Backup -памяти в байтах, выделенного для NVLib.

int HAL_bram_write(int n, int *src, int offset);

Запись <n> байт из буфера <src> в Backup-память со смещением <offset>. Количество записываемых данных должно быть кратно четырем байтам. Адрес буфера и смещение должны быть выровнены на границу четырех байт. Размер записываемых данных и смещение должны быть выбраны таким образом, чтобы исключить выход за пределы сегмента. При успешном выполнении возвращается «0», иначе код ошибки:

GNSS_BAD_PARAM – неправильные параметры;

GNSS_ERROR – ошибка записи.

int HAL_bram_read(int n, int *dst, int offset);

Чтение <n> байт из Backup-памяти со смещением <offset> в буфер <dst>. Количество записываемых данных должно быть кратно четырем байтам. Адрес буфера и смещение должны быть выровнены на границу четырех байт. Размер записываемых данных и смещение должны быть выбраны таким образом, чтобы исключить выход за пределы сегмента. При успешном выполнении возвращается «0», иначе код ошибки:

GNSS_BAD_PARAM – неправильные параметры;

GNSS_ERROR – ошибка чтения.

4.3.2.2. Доступ к часам реального времени:

int HAL_rtc_set(longlong time);

Установка таймера реального времени. В случае успешной установки возвращается «0». В случае ошибки:

GNSS_REGECTED – платформа отказалась устанавливать время;

GNSS_ERROR – прочие ошибки.

int HAL_rtc_get(longlong &time);

Чтение таймера реального времени. Возвращаемое значение: число мс от <>. При успешном выполнении возвращается «0», иначе:

GNSS_INVALID – возвращаемое время невалидно;

GNSS_ERROR – прочие ошибки.

4.4. Интерфейс DBG-HAL

Интерфейс DBG-HAL является абстракцией прикладного символьного интерфейса. Поставщиком интерфейса является отладочный модуль DBG библиотеки NVLib.

4.4.1. Функции интерфейса DBG-HAL

4.4.1.1. В приложениях, где используются данные в формате NMEA, данный интерфейс может быть использован в качестве информационно-командного.

int GNSS_dbg_rx(int n, char *buf);

Назначение: передача отладочному модулю NVLib n символов строки или части строки команды.

Период вызова: по необходимости.

Прерываемость: да.

Время выполнения: 0-100 мкс.

Функция выполняется в кванте времени вызывающего процесса (HAL или приложения) и осуществляет разбор команды управления и непосредственное выполнение простых команд. Непосредственное исполнение сложных команд выполняется при очередном вызове **`GNSS_solve()`**.

При успешном выполнении возвращает значение, большее нуля.

void GNSS_set_dbg_tx_callback(GNSS_dbg_tx_callback_f callback_func)

Назначение: установка указателя функции обратного вызова GNSS_dbg_tx_callback_f для передачи сообщений из отладочного модуля NVLib.

Период вызова: функция должна вызываться из HAL или приложения однократно при инициализации.

typedef void (GNSS_dbg_tx_callback_f*)(int n, char *buf);

Назначение: функция обратного вызова для передачи сообщений из отладочного модуля NVLib в последовательный порт. Должна определяться в HAL или приложении и передаваться в NVLib с помощью функции GNSS_set_dbg_tx_callback.

Период вызова: функция вызывается из GNSS_solve в соответствии с установленной скоростью.

Прерываемость: да.

Время выполнения (максимальное): не должно превышать 1 мс.

Функция должна переписать передаваемые данные в буфер передачи последовательного порта и немедленно вернуться.

4.5. Интерфейс NVL-APP

4.5.1. Функции интерфейса NVL-APP

4.5.1.1. Функции данного интерфейса предназначены для установки параметров навигационного ядра и получения информации о координатах, времени и текущем статусе приемника.

int GNSS_send_cmd(GNSS_cmd_head *cmdParam);

Назначение: функция для посылки команды управления.

Период вызова: по необходимости.

Прерываемость: да.

Время выполнения: 0-100 мкс на частоте RISC 200 МГц.

Функция выполняется в кванте времени вызывающего процесса и осуществляет разбор команды управления и непосредственное выполнение простых команд. Непосредственное выполнение сложных команд выполняется при очередном вызове **`GNSS_solve()`**.

При успешном выполнении возвращает «0».

```
typedef struct {
  int size; //размер буфера данных в 32-битных словах, включая поле size
  GNSS_cmd_code cmd; //код команды
} GNSS_cmd_head;
```

Назначение: заголовок дескриптора команды управления, используемый в качестве параметра функции GNSS_send_cmd. Дескрипторы команд определяются отдельно для каждой команды и должны первым полем включать ее заголовок.

Размер дескриптора определяется полем size. Управление предполагает задание режимов работы NVLib, таких как используемая система (GPS, GLONASS, GPS+GLONASS), темп формирования решений, перезапуск и т.п.

void GNSS_set_rmc_callback(GNSS_rmc_callback_f func)

Назначение: установка указателя функции обратного вызова **GNSS_rmc_callback_f** для приема минимальных навигационных параметров.

Период вызова: должна вызываться ОС или приложением однократно при инициализации.

typedef void (GNSS_rmc_callback_f*)(GNSS_rmc_msg *msg, int period);

Назначение: функция обратного вызова для приема минимальных навигационных параметров. Должна определяться в ОС или приложении и передаваться в библиотеку с помощью функции GNSS_set_ rmc_callback.

Период вызова: функция вызывается из GNSS_solve в соответствии с установленной скоростью.

Прерываемость: да.

Время выполнения: не более 100 мкс.

```
typedef struct {
  long   Valid;        //валидное решение
  long   time;         //текущее время
  long   lat_int;      //широта (целая часть)
  long   lat_frac;     //широта (дробная часть)
  long   lon_int;      //долгота (целая часть)
  long   lon_frac;     //долгота (дробная часть)
  long   alt;          //высота
  long   DOP;          //DOP
  long   tot_sat;      //количество наблюдаемых спутников
  long   sol_sat;      //количество спутников в решении
} GNSS_rmc_msg;
```

Назначение: формат сообщений, передаваемых функции GNSS_rmc_msg в качестве параметра. Содержит минимально необходимые навигационные параметры.

4.6. Команды управления

```
typedef enum{
  GNSS_restart,
  GNSS_set_mode,
  GNSS_get_mode,
  GNSS_set_rate,
  GNSS_get_rate,
  GNSS_set_sat_filter,
  GNSS_get_sat_filter,
  GNSS_set_sat_mask,
  GNSS_get_sat_mask
} GNSS_cmd_code;
```

4.6.1. GNSS_restart

4.6.1.1. Назначение: рестарт навигационной задачи.

Формат:

```
typedef enum{
  GNSS_cold,
  GNSS_warm,
  GNSS_hot,
  GNSS_reaquisition
} GNSS_restart_code;

typedef struct {
  GNSS_cmd_head head;
  GNSS_restart_code;
} GNSS_cmd_restart;
```

4.6.2. GNSS_set_mode, GNSS_get_mode

4.6.2.1. Назначение: установка/возврат установки режима работы.

Формат:

```
typedef enum{
  GNSS_GPS,
  GNSS_GLONASS,
  GNSS_GPS_GLONASS,
} GNSS_sat_system;

typedef struct {
  GNSS_cmd_head head;
  GNSS_sat_system system;
} GNSS_cmd_mode;
```

4.6.3. GNSS_set_rate, GNSS_get_rate

4.6.3.1. Назначение: установка/возврат установки скорости выдачи решения (вызова GNSS_solve).

Формат:

```
typedef struct {
  GNSS_cmd_head head;
  int rate_ms;
} GNSS_cmd_rate;
```

4.6.4. GNSS_set_sat_filter, GNSS_get_sat_filter

4.6.4.1. Назначение: установка фильтрации спутников.

Формат:

```
typedef struct {
  GNSS_cmd_head head;
  int angle;      //минимальный возвышения в градусах
  int snr;        //минимальный SNR в дБ
} GNSS_cmd_sat_filter;
```

4.6.5. GNSS_set_sat_mask, GNSS_get_sat_mask

4.6.5.1. Назначение: установка маски спутников.

Формат:

```
typedef struct {
  GNSS_cmd_head head;
  GNSS_sat_system sytem;        //навигационная система GPS или ГЛОНАСС
  int32 track_msk;              //маска спутников в слежении (-1: все спутники)
  int32 solve;                  //маска спутников в решении (-1: все спутники)
} GNSS_cmd_sat_mask;
```

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

ГЛОНАСС – ГЛОбальная НАвигационная Спутниковая Система

ОС – операционная система

ПО – программное обеспечение

ПЭВМ – персональная электронно-вычислительная машина

GPS – Global Positioning System

RISC – Reduced Instruction Set Computer

Лист регистрации изменений

Изм.	Номера листов (страниц)				Всего листов (страниц) в документе	№ документа	Входящий № сопроводительного документа и дата	Подпись	Дата
	измененных	замененных	новых	аннулированных					