Встраиваемое программное обеспечение «Расширение AT команд» для многофункционального модуля GSM/ГЛОНАСС TE-SL6087-NV08C

Руководство пользователя

Версия:	*1.2*
Дата:	*2011/05/18*

ОГЛАВЛЕНИЕ

1 Введение

Встраиваемое программное обеспечение «Расширение AT команд» для многофункционального модуля GSM/ГЛОНАСС TE-SL6087-NV08C реализует набор специализированных AT команд для настройки и управления навигационным приёмником NV08C, входящим в состав TE-SL6087-NV08C, а также для получения и обработки навигационных данных.

Функционал программы включает одометр, контроль скорости, работу с геозонами.

«Расширение AT команд» представляет собой приложение для операционной системы Open AT от Sierra Wireless.

Open AT – это операционная система, используемая в GSM модулях Sierra Wireless. Она включает в себя реализацию всех GSM функций (передачу данных, голоса, стандартные AT команды и т.п.) и кроме этого предоставляют программные сервисы «обычных» (не-GSM) операционных систем. В результате у разработчика появляется возможность выполнять своё приложение прямо на GSM модуле, на том же процессоре, где выполняется стек протоколов GSM.

«Расширение AT команд» доступно полностью в открытых исходных кодах, что позволяет Вам расширять и адаптировать его под свои конкретные нужды, и использовать в качестве отправной точки для разработки Вашего собственного навигационного приложения.

В последней главе этого документа Вы узнаете, как превратить Ваш модуль TE-SL6087-NV08C в полноценный ГЛОНАСС/GPS/GSM трекер за 30 минут (или меньше).

2 Функции встраиваемого ПО «Расширение AT команд»

Этот раздел содержит описание всех AT команд, реализуемых приложением.

2.1 Получение навигационных данных

2.1.1 Запрос навигационных данных

Формат:

AT+GNPOS=p,d,v,a,h,c

Ответ модема:

```
GNPOS=<широта>,<долгота>,<дата>,<время>,<скорость>,<азимут>,<высота>,<СКО>,
<валидность>
OK
```

Описание:

Поля в строке запроса имеют следующие значения:

P – координаты (широта и долгота)

D – дата и время

V – скорость

A – азимут

H – высота

C – СКО, PDOP

По данной команде выдаётся текущая позиция, полученная из навигационного приёмника (НП) на момент отправки запроса. Поля в строке ответа соответствуют полям в строке запроса. Если в запросе какой-то из параметров не указан (пропущен), то соответствующий параметр не выдаётся и в ответе. Например, для того, чтобы получить только текущую координату и значение азимута,

необходимо выполнить команду `AT+GNPOS=p,,,a`. Ответом на данную команду, в случае наличия достоверной координаты, будет `GNPOS=<широта>,<долгота>,,,,<азимут>,,,Y`.

Широта передается в градусах с десятичными долями. Диапазон значений от 0 до 90. Например: 55.657433

Долгота передается в градусах с десятичными долями. Диапазон значений от 0 до 180. Например: 37.342546

Полушарие задается знаком. Широта положительная в северном полушарии и отрицательная в южном. Долгота положительная в восточном полушарии и отрицательная в западном. В качестве разделителя дробной части используется точка. Точность, количество знаков в дробной части, не регламентируется и зависит от реализации.

Скорость передается в км/ч.

Азимут передается в градусах по часовой стрелке от севера. Диапазон значений от 0 до 360. Представляется в виде целого числа.

Дата передается в формате `dd.mm.yyyy`.

Время передается в формате `hh.mm.ss`. Часовой пояс по UTC.

Высота передается в метрах, целое.

Валидность передается одной буквой `Y` или `N`. Поле передается всегда, вне зависимости от строки запроса.

2.1.2 Запрос видимых спутников

Формат:

AT+GNSAT?

Ответ модема:

```
GNSAT=<данные по спутникам>
OK
```

Описание:

Возвращаются данные о спутниках, полученные от НП.

Формат строки:

(U, S, L, SN, A, H) повтор n раз

U – признак «использован в решении», буква Y или N

S – код системы, цифра: 1 – GPS, 2 – GLONASS, 3 – GALILEO, 4 – COMPASS, 5 – SBAS

L – литера спутника (номер), целое

SN – (signal/noise) – отношение сигнал/шум, целое

A – азимут

H – возвышение над горизонтом

n – количество наблюдаемых спутников

2.2 Одометр

2.2.1 Сброс одометра

Формат:

AT+GNODRES

Ответ модема:

```
OK
```

Описание:

Команда сбрасывает значение счетчика пробега.

2.2.2 Чтение одометра

Формат:

```
AT+GNODGET?
```

Ответ модема:

```
ODGET = пробег в метрах
OK
```

Описание:

Команда возвращает значение счетчика пробега в виде целого числа.

2.3 Контроль скорости

2.3.1 Сообщение модема о превышении скорости

Выдается периодически, если скорость превышает установленный порог, не равный 0. При пороге, равном 0 контроль скорости не ведется.

Формат сообщения:

```
GNOVERSPEED= скорость в км/ч, широта, долгота, дата, время
```

2.3.2 Установка порогового значения скорости

Формат:

```
AT+GNOVERSPEED = скорость в км/ч
```

Ответ модема:

```
OK
```

Описание:

Команда устанавливает порог скорости, превышение которого будет считаться нарушением и приведет к выдаче сообщения.

2.4 Геозоны

Геозоны поддерживаются в двух видах – круг и прямоугольник. Контроль входа и выхода устройства из зоны ведется периодически с получением новых данных о местоположении. Контроль осуществляется только за зонами, которые включены.

2.4.1 Сообщение от модема при входе в (выходе из) зоны

```
GNGFIN = номер зоны1,…,номер зоны N
GNGFOUT = номер зоны1,…,номер зоны N
```

GNGFIN выдает список зон, в которые объект вошел и находится внутри.

GNGFOUT выдает список зон, из которых объект вышел.

Примечания:

- Номера зон, в которые объект не входил, не выдаются.

- Проверка ведется только для активных геозон.

2.4.2 Добавление геозоны

Формат:

Для круговой зоны:

AT+GNGFADD=n,c,x,y,r,S

Для прямоугольной зоны:

AT+GNGFADD=n,r,x2,y2,x2,y2,S

Ответ модема:

OK

При ошибке:

ERROR n – если описание зоны с номером n содержит ошибку

Ошибкой считается:

- повторение номеров зон,
- тип, отличный от c или r
- неверные координаты (не удалось разобрать как действительные числа)
- недостаточно параметров

Описание:

n – номер зоны. Должен быть уникальным.

C – тип зоны круговая,

R – тип зоны прямоугольная

Для круговой зоны:

X, Y – координаты центра зоны, действительные в градусах

R – радиус зоны, целое число в метрах.

Для прямоугольной зоны:

X1, Y1 – координаты левого верхнего угла зоны (северо-западного), действительные в градусах

X2, Y2 – координаты правого нижнего угла зоны (юго-восточного), действительные в градусах

S – активная (A) или пассивная (P).

2.4.3 Запрос состояния геозон

Формат:

AT+GNGF?

Ответ модема:

GNGF = <список геозон>
OK

Описание:

Выдаётся список геозон, состоящий из последовательностей N, T, S, P для заданных зон, где:

N – номер зоны

T – тип (C или R)

S – состояние (A – активна, P – пассивна, отключена)

P – положение относительно нее (I – внутри, O – снаружи)

2.4.4 Изменение состояния геозоны

Формат:

```
AT+GNGFSET=n,A|P
```

Ответ модема:

OK	Если команда принята и выполнена
ERROR	Если какие-то из параметров заданы неверно (нет геозоны с номером *n*, или второй параметр – не один из символов *A* или *P*)

Описание:

Устанавливает геозону с номером *n* в активное (*A*) или пассивное (*P*) состояние.

2.4.5 Удаление геозоны

Формат:

```
AT+GNGFDEL=n
```

Ответ модема:

OK	Если команда принята и выполнена
ERROR	Если параметр *n* задан неверно (нет такой геозоны)

Описание:

Удаляет геозону с номером *n*.

2.5 Управление работой навигационного приёмника NV08C

2.5.1 Выбор используемой навигационной системы

Команда задает выбор используемой навигационной спутниковой системы для НП.

Формат:

```
AT+GNSS=<код>
```

Ответ модема:

OK	Если команда принята и выполнена
ERROR	Если параметр код задан неверно

Описание:

Параметр *код* принимает значения, исходя из следующей битовой маски:

ТИП ГНСС	COMPASS	GALILEO	GLONASS	GPS
Значение бита (0/1)	0	0	X	X

Параметр	Двоичный код	Описание/ используемые сигналы
1	0001	GPS
2	0010	GLONASS
3	0011	GPS + GLONASS

Примечание:

Несмотря на то, что системы COMPASS и GALILEO в настоящее время не поддерживаются НП, установка соответствующих им битов в маске в 1 ошибкой не считается. Эти биты просто игнорируется.

2.5.2 Включение дифференциальных поправок SBASS

Формат:

AT+GNSBAS=<ON|OFF>

Ответ модема:

При успешно выполненной команде

```
GNSBASS=<ON|OFF>
OK
```

При ошибочно введенной команде

```
ERROR
```

Ошибка возникает в случае ввода параметра, отличного от ON или OFF.

Описание:

Команда включает и выключает прием сигналов дифференциальных поправок SBAS.

2.5.3 Отключение питания

Формат:

AT+GNPWROFF

2.5.4 Установка режима энергопотребления

Формат:

AT+GNPOWERON=BASE|LOW|TTF

Описание:

BASE – базовый режим работы (включен)

LOW – режим пониженного потребления

TTF – Time-to-Time Fix – режим периодического приема сигналов.

2.5.5 Задание параметров работы НП

Команда требуется для начальной инициализации НП.

Формат:

AT+GNPAR=DEFAULT|угол возвышения спутника, предельное СКО (PDOP), отношение сигнал-шум

Ответ модема:

```
OK
```

Описание:

Угол возвышения спутника – минимальный угол над горизонтом, ниже которого спутник не участвует в определении местоположения (целое).

Предельное значение СКО (среднеквадратичного отклонения) – это величина отклонения, не превышая которое, решение считается валидным (целое).

Отношение сигнал-шум (SNR) – минимальное значение отношения полезного сигнала к шуму (signal to noise rate), при котором данные со спутника ещё учитываются при решении навигационной задачи.

Заданные настройки сохраняются в НП.

При значении параметра DEFAULT НП дается команда перейти к заводским установкам.

2.5.6 Сброс НП

Формат:

AT+GNRESET=SOFT|FULL

Описание:

`SOFT` – программный сброс приемника без стирания данных («теплый» старт)

`FULL` – сброс со стиранием данных («холодный» старт).

2.5.7 Конфигурирование UART A

Формат:

Установка параметров:

AT+GNUARTA=P,B

Описание:

Команда для конфигурирования порта UART A НП (порт UART B используется для обмена с контроллером SL6087):

P – протокол обмена (может принимать значение NMEA, BINR, RTCM)

B – скорость обмена (может принимать значение 4800, 9600, 19200, 38400, 57600, 115200, 230400).

Запрос состояния:

AT+GNUARTA?

Ответ модема:

```
GNUARTA=P,B
```

Описание:

P – протокол обмена (NMEA, BINR, RTCM), строковое

B – скорость обмена (целое число).

3 Работа с исходным кодом, расширение функциональности

В этой главе Вы найдёте краткие инструкции о том, как начать работать с исходным кодом «Расширения AT команд», как перекомпилировать программу и перепрограммировать Ваш модуль TE-SL6087-NV08C.

Информация о внутреннем устройстве, программной архитектуре этого приложения – в следующей главе.

3.1 Установка Sierra Wireless Developer Studio

Первым техническим шагом в программировании для Open AT является установка среды разработки – Sierra Wireless Developer Studio. Она доступна для скачивания на сайте `www.sierrawireless.com`.

Установите среду разработки, следуя инструкциям на сайте Sierra Wireless. В процессе установки программа-установщик задаст Вам вопрос о том, какие программные компоненты необходимо установить. Кроме самого Developer Studio, Вам обязательно надо выбрать для установки следующие компоненты:

- ARM ELF GCC Toolchain
- Open AT OS Package
- Firmware Package
- WIP Plug-in Package.

На запрос программы-установщика **Please select your product family** следует выбрать ответ **AirPrime SL Series EDGE**.

3.2 Установка исходных кодов «Расширения AT команд»

1. Распакуйте архив с программным обеспечением `sl6087-nv08c-at-extensions.zip` во временную папку.
2. В Sierra Wireless Developer Studio выберите пункт меню **File->Import**.
3. В открывшемся окне выберите **Developer Studio, Existing Developer Studio project**; нажмите **Next**.
4. В открывшемся окне **Select root directory** выберите ту директорию, в которую Вы на первом шаге распаковали архив с исходным кодом; нажмите **Finish**.

3.3 Компиляция/сборка проекта

Для компиляции проекта выделите левой кнопкой мыши директорию с проектом в окне Project Explorer и выберите команду меню **Project -> Build project**.

По окончании процесса сборки исполняемые файлы будут размещены в поддиректории `[Target]_ARM_ELF_GCC_Release` в дереве проекта:

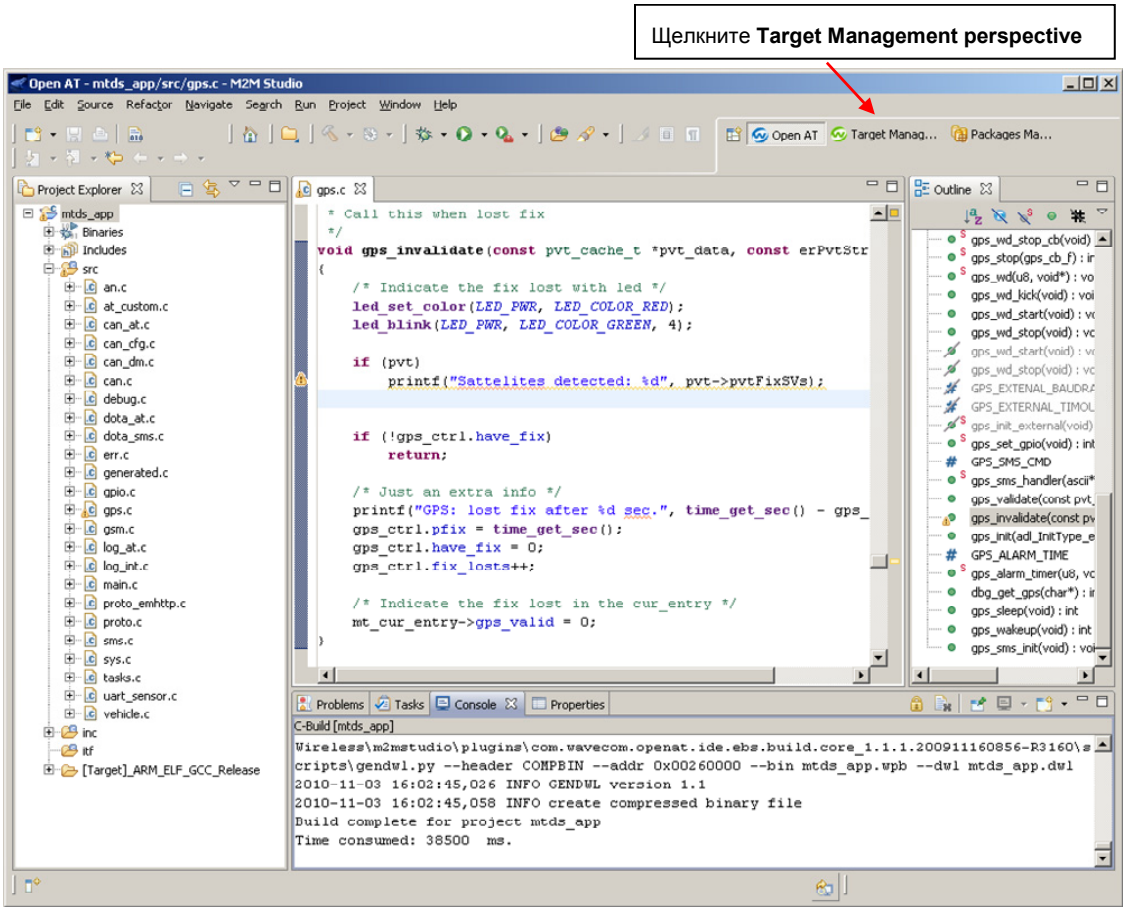
- `<имя проекта>.wpb` - для программирования с помощью DwlWin;
- `<имя проекта>.dwl` - для программирования с помощью **`AT+DWL`** команды или средствами Developer Studio.

3.4 Обновление прошивки SL6087

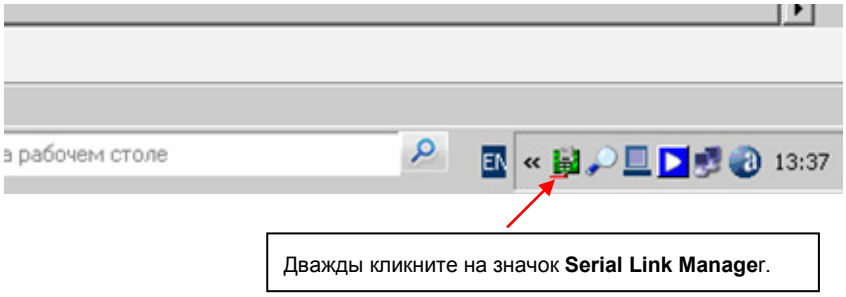
Для перепрограммирования (прошивки) TE-SL6087-NV08C новой версией встроенного приложения, выполните следующие шаги:

1. Выполните компиляцию/сборку Вашего приложения, как описано в предыдущем разделе.
2. Подключите TE-SL6087-NV08C к компьютеру.
3. Если у Вас открыто окно **HyperTerminal**, подключённое к TE-SL6087-NV08C, то закройте его.

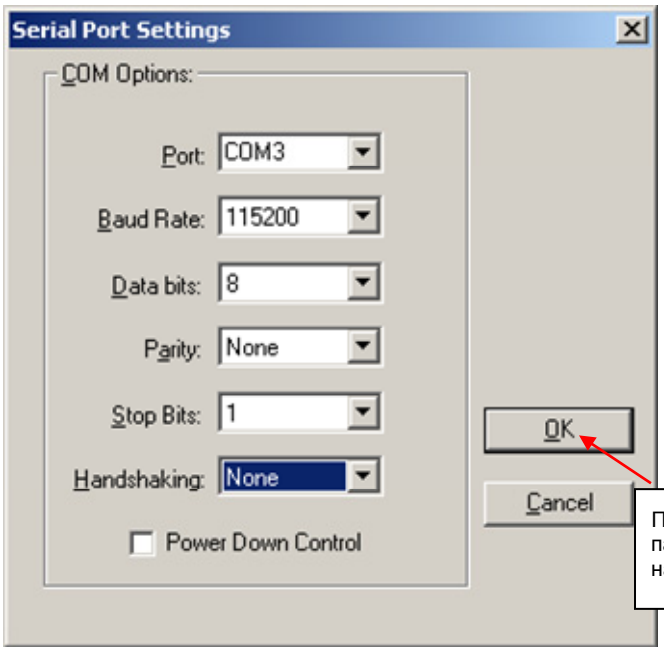
4. В M2M Studio перейдите в раздел **Target Management**, нажав кнопку **Target Management perspective** в правом верхнем углу основного окна.



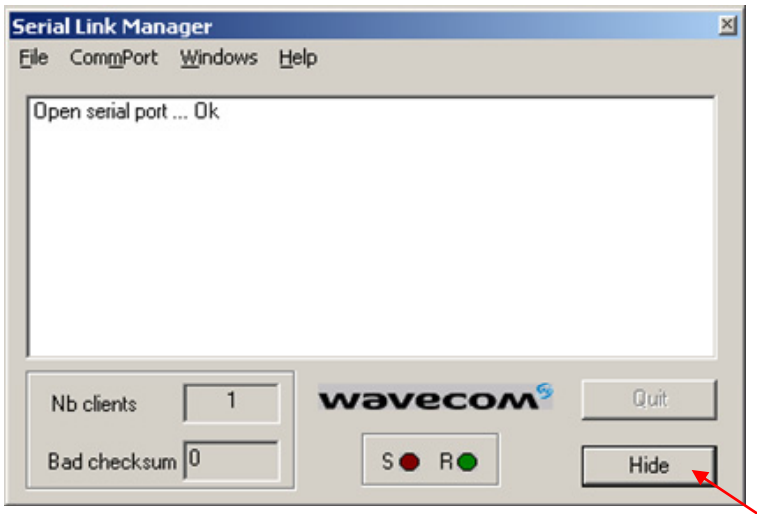
5. В **System Tray** дважды кликните на значок **Serial Link Manager**.



6. В открывшемся окне **Serial Link Manager** воспользуйтесь командой меню **CommPort -> Settings**. Установите следующие параметры: **Port**: COM*n*, соответствующий COM порту компьютера, к которому подключен Ваш модуль; **Handshaking**: None, **Stop Bits**: 1, **Parity**: None, **Data bits**: 8, **Baud** Rate: 115200. Нажмите **OK**. В окне **Serial Link Manager** нажмите **Hide**.

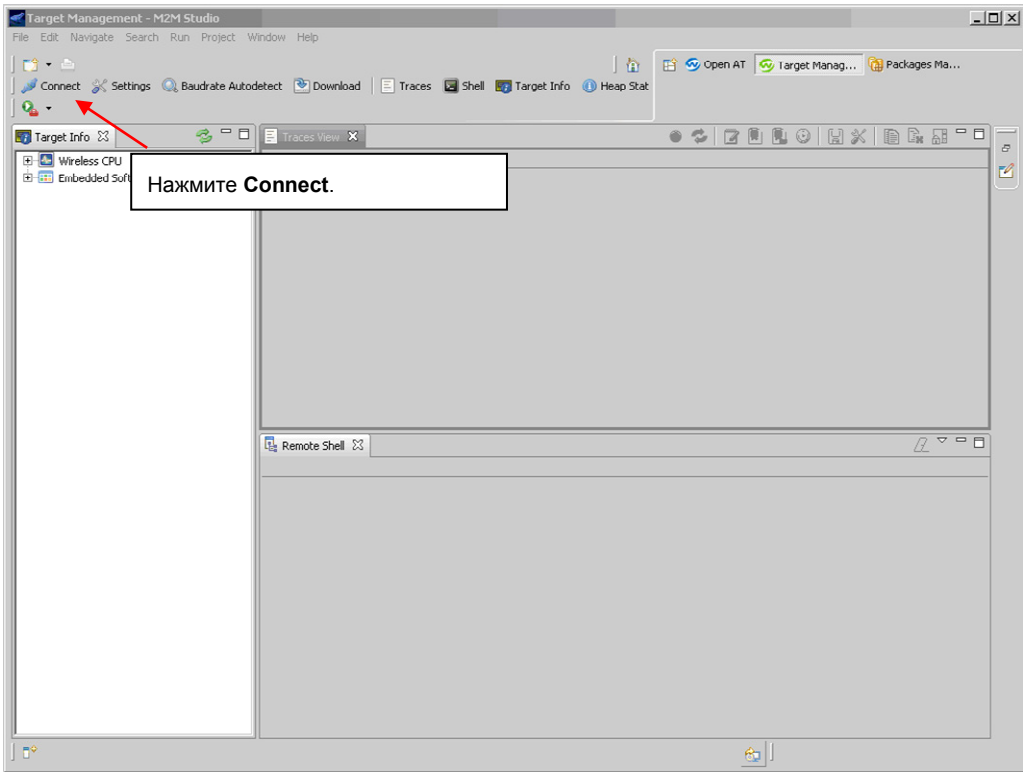


После установки параметров нажмите **OK**.

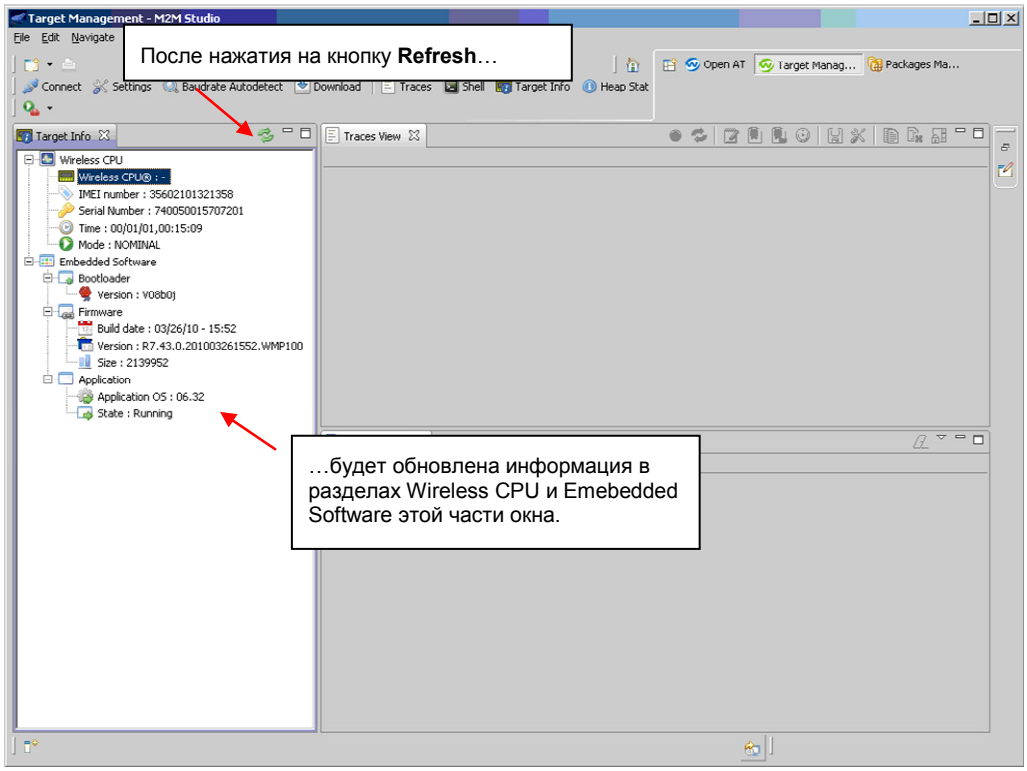


Нажмите **Hide**.

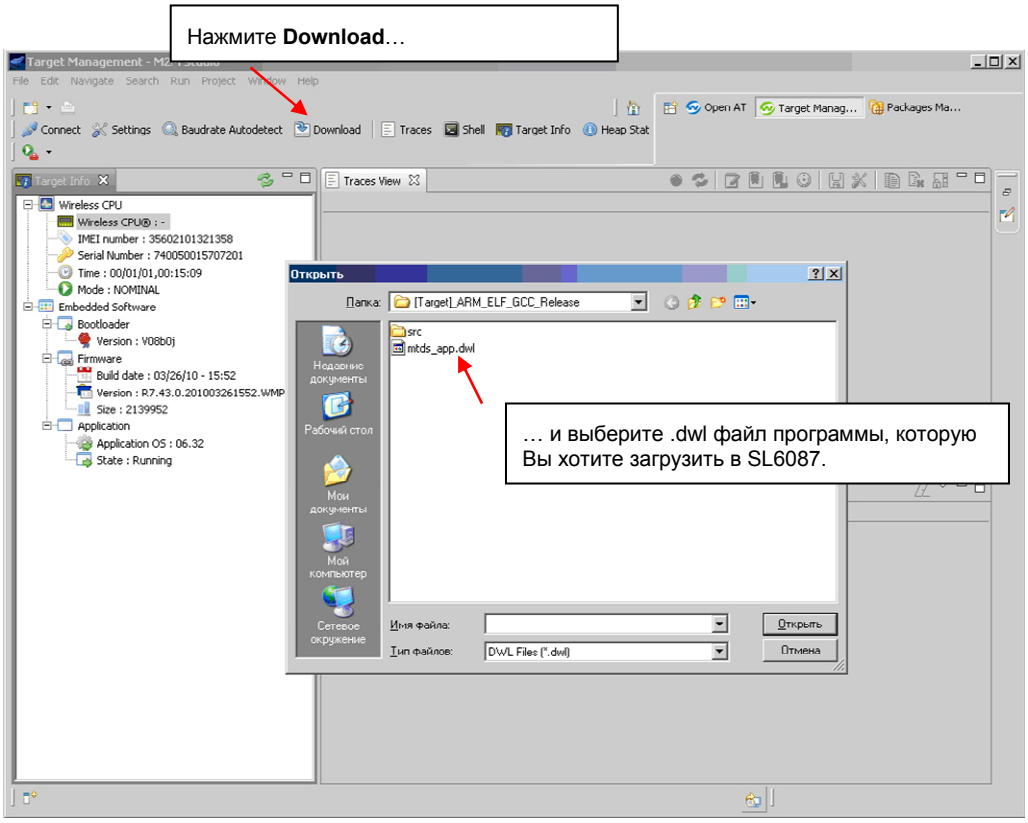
7. В M2M Studio нажмите кнопку **Connect**.



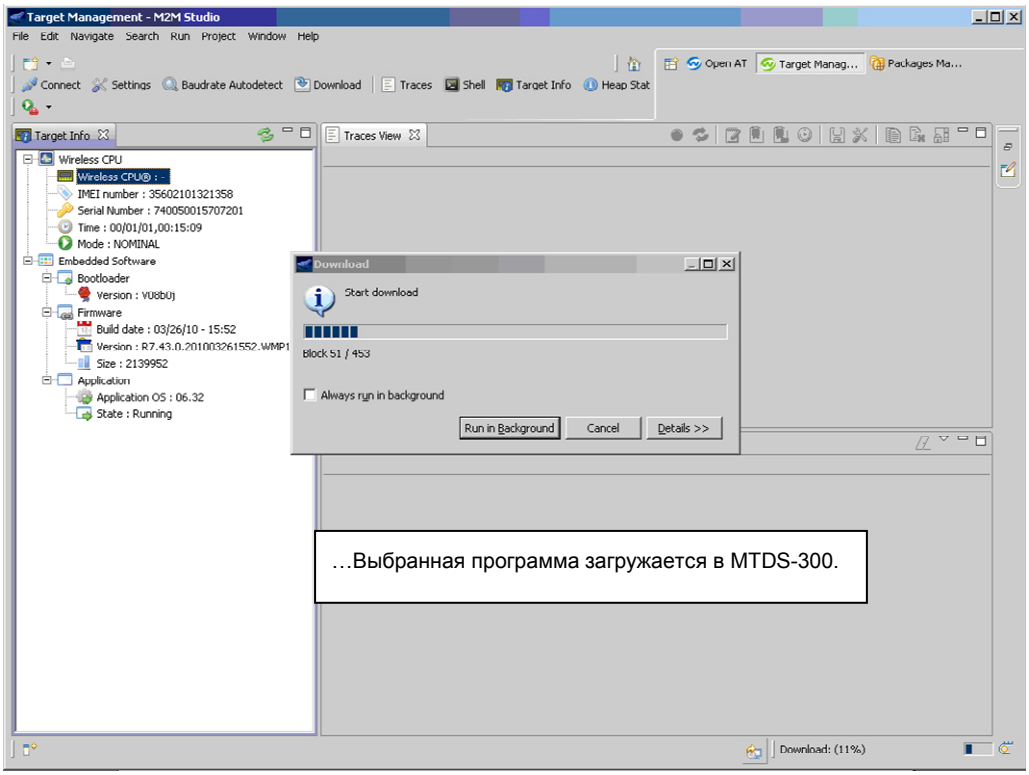
8. Убедитесь, что подключение прошло успешно, нажав кнопку **Refresh** в правой верхней части раздела **Target Info**. После нажатия на эту кнопку, в случае успешного подключения, будет обновлена соответствующая информация в разделах **Wireless CPU** и **Emebedded Software** этой части окна.



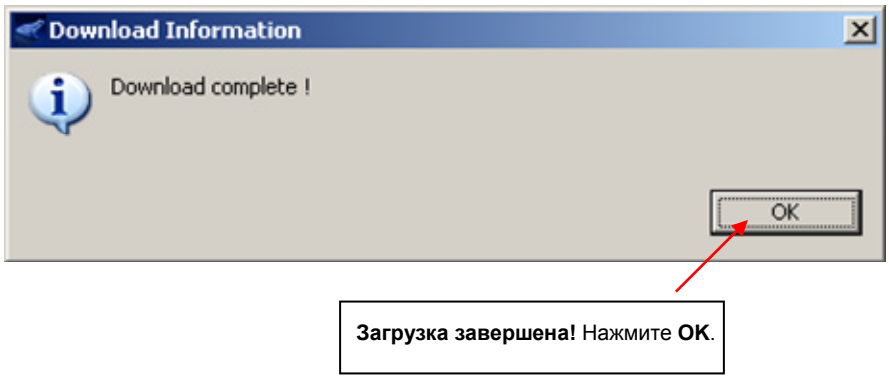
9. Нажмите кнопку **Download** на панели инструментов и выберите файл `.dwl` программы, которую Вы хотите загрузить в TE-SL6087-NV08C.



10. В появившемся окне **Download** M2M Studio отражает текущее состояние процесса перепрограммирования TE-SL6087-NV08C Вашим файлом.



При успешном завершении процесса появляется окно **Download complete ! -** нажмите в нём кнопку **OK**.



Загрузка завершена! Нажмите **OK**.

После программирования процессорного модуля приложением «Расширение AT команд», при старте программы выдаётся строка в консоль, содержащая дату и время сборки приложения; например:

```
NV08C-DK v1.00 Feb 20 2011 13:20:09
```

4 Программная архитектура «Расширения AT команд»

В этом разделе описывается внутреннее устройство приложения.

4.1 Подсистемы приложения

«Расширение AT команд» состоит из четырёх взаимодействующих друг с другом программных подсистем, каждая из которых реализована в отдельном потоке выполнения (Open AT Task). Вот перечень этих подсистем-потоков приложения в порядке убывания их приоритета:

- BINR
- NAVI
- CTRL
- UART.

Кроме перечисленных выше потоков выполнения, «Расширение AT команд» использует ещё один поток, который называется ADL. Это наиболее приоритетный поток, в рамках которого выполняются вызовы Open AT ADL API. В контексте этого же потока работает WIP плагин.

Поток BINR, взаимодействуя с подсистемой UART, реализует приём и передачу навигационных данных по протоколу BINR.

Подсистема NAVI осуществляет математическую обработку поступающей навигационной информации для реализации функциональности одометра, контроля скорости и геозон, взаимодействует с подсистемами BINR и CTRL.

В рамках подсистемы CTRL регистрируются реализуемые приложением AT команды. Эта подсистема взаимодействует с BINR и NAVI.

UART – это поток низкоуровневого обмена данными с навигационным приёмником через последовательный порт GSM модуля. Эта подсистема использует Open UART Interface Open AT.

4.2 Принципы работы и взаимодействия подсистем

4.2.1 Общая концепция

- Инициаторами BINR обмена являются подсистемы CTRL и NAVI;
- Подсистемы CTRL и NAVI регистрируют свои функции обратной связи в BINR подсистеме, которая вызывает эти функции по поступлению указанных BINR пакетов;
- Функции обратной связи CTRL и NAVI выполняют обработку поступающих навигационных данных и в случае необходимости инициируют вывод соответствующей информации на консоль.

4.2.2 Последовательность событий при старте программы

Запуск осуществляется в порядке приоритетов задач, в контексте которых работают подсистемы (ADL, UART, BINR, NAVI, CTRL):

- ADL инициализирует wipSOFT и больше явным образом не вызывается;
- BINR инициализирует машину состояний и выполняет while(1) засыпая до возникновения запросов на передачу со стороны CTRL/NAVI, либо оповещения о приёме со стороны UART; распределение поступающих BINR-пакетов в другие подсистемы (NAVI и CTRL) централизовано реализуется именно в этом потоке;
- NAVI подписывается на необходимые BINR ответы, выполняет while(1), отправляет BINR - запросы, засыпая либо до поступления BINR-ответов, либо до истечения таймаута; события приёма генерируются из функции обратной связи вызываемой BINR по поступлению BINR

response, на который подписан NAVI; данные в этой же функции обратной связи из пакетов помещаются во внутренние структуры, обрабатываемые после пробуждения потоком NAVI;

- CTRL подписывается на необходимые BINR -ответы, подписывается на **AT+** команды, далее выполняет while(1), засыпая до возникновения необходимости отправки BINR-запросов, которая инициируется обработчиками **AT+** команд от пользователя; функции обратной связи, вызываемые по поступлению BINR-пакетов извлекают из пакетов необходимую информацию и выводят её в консоль в качестве ответа на **AT+** команду (в т.ч. и просто OK, если **AT+** команда не запрашивает каких-либо данных, а выполняет какую-то настройку NV08C чипа; например, - выбор навигационной системы);
- UART после подготовительных действий выполняет while(1) засыпая до возникновения событий приёма/передачи;

4.2.3 Подсистема UART

Эта подсистема реализует следующий API:

```
int uart_send(char *buf, unsigned int len);
int uart_async_rx_reg(void (*cb)(char *data, int len));
```

При старте задачи настраивается Open UART Interface и выполняется ожидание появления событий передачи/приёма.

Передача инициируется из BINR подсистемы, выполняющей вызов uart_send().

Поскольку UART задача является наименее приоритетной, то и стартует она последней. Это означает, что более высокоприоритетные задачи (в нашем случае NAVI) могут инициировать binr_send() -> uart_send() в тот момент, когда UART задача ещё не была запущена и проинициализирована. В этой ситуации в uart_send() выполняется останов высокоприоритетной задачи, слишком рано инициировавшей отправку в UART, с помощью adl_ctxSuspend(). После старта и завершения инициализации UART задачи выполнение приостановленных высокоприоритетных задач возобновляется с помощью вызова adl_ctxResume().

Интерфейс приёма – асинхронный: BINR, с помощью uart_async_rx_reg(), регистрирует свой обработчик, который, вызываясь из UART подсистемы, перекладывает принятые данные в свой внутренний буфер и активизирует BINR поток (BINR машину состояний).

4.2.4 Подсистема BINR

Эта подсистема реализует следующий API:

```
/*
 * Supported BINR commands identifiers
 */
enum binr_cmd_id {
        BINR_CMD_01             = 0x01,
        BINR_CMD_0B             = 0x0B,
        BINR_CMD_0D             = 0x0D,
        ...
};

/*
 * Supported BINR responses identifiers
 */
enum binr_rsp_id {
        BINR_RSP_UNKN           = 0x00,
        BINR_RSP_41             = 0x41,
        BINR_RSP_46             = 0x46,
        BINR_RSP_50             = 0x50,
        ...
};
```

```
/*
 * BINR command structures
 */
struct binr_cmd_01 {
        u8                      c0;
        u8                      c1;
        u8                      c2;
        u8                      c3;
        u8                      c4;
        u8                      prm;
} __attribute__((packed));

struct binr_cmd_0b {
        u8                      port;
        u32                     baud;
        u8                      prot;
} __attribute__((packed));
struct binr_cmd_...;

/*
 * BINR response structures
 */
struct binr_rsp_unkn {
        u8              prm[BINR_PKT_MAX_LEN]; /* see BINR, ch.5.1 */
} __attribute__((packed));

struct binr_rsp_41 {
        float                   head;
        float                   speed;
        u32                     week_sec;
} __attribute__((packed));

struct binr_rsp_46 {
        u32                     week_sec;
        u8                      day;
        u8                      month;
        u16                     year;
        s8                      dhour;
        s8                      dminute;
} __attribute__((packed));

struct binr_rsp_...;

/*
 * Descriptor of BINR response packets to subscribe to
 */
struct binr_dsc {
        enum binr_rsp_id        id;
        void                    (*cb)(enum binr_rsp_id id,
                                      void *rsp,
                                      int  len);
};

/*
 * Functions
 */
int binr_cmd_send(enum binr_cmd_id id, void *data, int dlen);
int binr_rsp_reg(enum binr_rsp_id id,
                 void (*cb)(enum binr_rsp_id id, void *rsp, int len));
```

Приём реализован в виде машины следующих состояний (отрабатывается в основном цикле потока):

- <PASSIVE>: BINR подсистема неактивна, ожидает события binr_act;

В любом другом состоянии BINR подсистема активна; ожидает событий binr_deact и uart_event_rx; по возникновению uart_event_rx выполняется побайтное чтение из UART-подсистемы:

<IDLE>	▪ при поступлении любого байта кроме DLE машина состояния не меняет и остаётся в <IDLE>; ▪ при поступлении DLE - инициализируется статический дескриптор нового пакета и выполняется переход в состояние <HEAD>.
<HEAD>	▪ при поступлении DLE, CRC, ETX - дескриптор нового пакета удаляется и выполняется переход обратно в состояние <IDLE>; ▪ полученный идентификатор ответа сохраняется в соответствующем поле дескриптора, переход в <DATA>.
<DATA>	▪ при поступлении любого байта кроме DLE - он помещается в буфер (содержится в дескрипторе), инкрементируется счётчик (размер сообщения), переход опять в <DATA>; ▪ при поступлении DLE - счётчик не инкрементируется, байт в буфер не помещается, переход в <END>.
<END>	▪ при поступлении DLE - байт помещается в буфер с данным, счётчик инкрементируется - переход обратно в <DATA> (сжатие DLE); ▪ при поступлении CRC - переход в состояние <PRE_CRC>; ▪ при поступлении ETX - переход в состояние <DONE>; ▪ при поступлении любого другого байта - игнорирование пакета (ошибка) и переход в <IDLE>.
<PRE_CRCx>	▪ считываются два байта CRC в дескриптор; ▪ третьим ожидается DLE - если пришёл он - переход в <POST_CRC>; ▪ если третьим пришёл не DLE - игнорирование пакета и переход в <IDLE>.
<POST_CRC>	▪ ожидается ETX, при его поступлении - переход в состояние <CHECK>; ▪ при поступлении любого другого - игнор пакета и переход в <IDLE>.
<CHECK>	▪ проверяется CRC - если корректная - переход в <DONE>; если нет - пакет отбрасывается и выполняется переход в <IDLE>.
<DONE>	▪ сюда попадаем, когда пакет полностью получен (и проверен, если присутствует CRC); передаём пакет в функции обратной связи; переход в <IDLE>.

По получению очередного пакета BINR выполняется его пост-обработка и последовательный вызов функций обратной связи, которые заинтересованы (подписывались с помощью binr_rsp_reg()) в получении данного пакета.

Пост-обработка заключается в преобразовании полей типа *FP64* (double) из VFP формата (стандарт IEEE 754, который используется в NV08C) в формат FPA, который используется в Open AT. Это преобразование заключается в перестановке местами слов 8-байтных double полей (в FPA старшее слово идёт первым независимо от “endianness” платформы). Теоретически, вместо этого преобразования можно было бы выполнять компиляцию приложения с флагами -msoft-float -mvfp вместо используемого сейчас флага -mfpa, но в этом случае не удаётся выполнить линковку: плагины и ADL собраны с флагом -mfpa. Подсистема BINR руководствуется идентификатором пакета для того, чтобы понять, есть ли в пакете FP64 поля и где они находятся.

Поля типа *FP32* (float) не преобразуются.

Поля типа *FP80* тоже не преобразуются, но только в виду того, что пока соответствующие поля таких пакетов нигде в приложении не используются. Если возникнет необходимость в их использовании,

то, вероятно, их придётся преобразовывать в double, т.е. с потерей точности, т.к. long double в Open AT имеет размер тех же 8 байт что и double, вместо ожидаемых для FP80 десяти байт.

Функция `binr_rsp_reg()` регистрирует функцию обратной связи (приёмного колбека) и работает с массивом `wm_lst_t rsp_cb[BIN_MAX_ID + 1]`, элементы которого указывают на списки функций, зарегистрированных на получение BINR-пакета с идентификатором равным индексу элемента в массиве. Размер этого массива – 1 КБ.

Следует отметить, что в функции обратной связи передаётся указатель на структуру типа `binr_rsp_unkn`, который впоследствии, на основании параметра id, может быть приведён функцией-обработчиком к указателю на одну из поддерживаемых структур (например, `binr_rsp_50`). Данные слов при этом искажаться не будут, так как процессор работает в режиме LSB (младшие байты по меньшим адресам).

Вызов `binr_cmd_send()` выполняется из `CTRL` или `NAVI`, в контексте вызывающего потока. С машиной приёма никак не конфликтует. Выполняет формирование пакета (экранирование символов и т.п.) и его отправку.

Обе функции (и приёма, и передачи) имеют дополнительный параметр: размер пакета. Он необходим, так как некоторые пакеты `BINR` имеют переменный размер и однозначно определить размер BINR пакета на основании только идентификатора BINR нельзя. Например, информация по спутникам присылается в виде пакета переменного размера, который содержит по фрейму заданного формата на каждый видимый спутник.

Как видно из API, в программном интерфейсе отсутствует функция де-регистрации приёмного колбека `BINR`-пакетов. Это сделано для упрощения: принимаемые пакеты всегда передаются на верхние уровни (`NAVI` и `CTRL`), а те уже, в свою очередь, решают, нужен ли им на данный момент этот пакет или не нужен.

4.2.5 Подсистема NAVI

Эта подсистема реализует следующий API:

```
/*
 * Odometer API
 */
int navi_odo_reset(void);
void navi_odo_show(void);

/*
 * Velocity API
 */
int navi_vel_set(int speed);

/*
 * Geozones API
 */
enum geo_state {
        GEO_STATE_ACTIVE,
        GEO_STATE_PASSIVE
};

int navi_geo_circ_add(int id, double x, double y, int r);
int navi_geo_rect_add(int id, double x1, double y1, double x2, double y2);
int navi_geo_state_set(int id, enum geo_state state);
int navi_geo_del(int id);
void navi_geo_show(void);
```

Подсистема реализует функциональность одометра, контроля скорости и контроля геозон.

При старте подписывается на получение необходимых `BINR response` пакетов.

В while(1) цикле, с ежесекундным интервалом (`adl_ctxSleep()`), выполняет выдачу соответствующих `BINR command` запросов.

Параметры контроля устанавливаются из подсистемы `CTRL` соответствующими вызовами.

4.2.5.1 Особенности контроля скорости

- Контроль не ведётся при пороге 0, а также при отсутствии валидной координаты;
- Сообщается о переходе через порог скорости (превышение, печатается текущая скорость);
- Сообщается о текущей скорости выше порога каждые `VEL_RATE` (~10 сек);
- Сообщается о переходе через порог скорости (ниже заданной, печатается полученная на предыдущей проверке (секунду назад) скорость, которая была ещё выше порога).

4.2.5.2 Особенности реализации одометра

- Ничего не делается при отсутствии валидной координаты, но при этом предыдущая координата не сбрасывается, то есть после восстановления валидной координаты (например, после выезда из туннеля), расстояние будет рассчитываться относительно той, до потери валидной координаты, точки;
- Ничего не делается при самом первом получении координаты;
- Ничего не делается, если координата не поменялась;
- Получается расстояние от текущей до предыдущей координаты; если оно меньше чем `ODO_PDOP_MULT * <PDOP>`, то предыдущая координата новой не обновляется, а в одометре это расстояние не учитывается; `<PDOP>` - текущее, полученное из NV08C, значение среднеквадратичного отклонения (ошибки), в метрах; по умолчанию множитель `ODO_PDOP_MULT` установлен в 2.

4.2.5.3 Особенности реализации геозон

- В случае отсутствия валидной координаты проверяется наличие геозон, которые отмечены как «назначенные на удаление» (`.type = GEO_TYPE_DELETE`); состояния геозон при этом не меняются (то есть остаются в прежних IN/OUT состояниях, что были до потери валидной координаты);
- Присутствие в геозоне-окружности определяется из сравнения расстояния от текущей координаты до центра окружности с её радиусом (если расстояние меньше или равно – значит, находимся внутри этой геозоны);
- Присутствие в геозоне-прямоугольнике определяется путём сравнения компонент текущей координаты с координатами прямых ограничивающих прямоугольник: `.lat`-компонента положительна на востоке и отрицательна на западе, растёт по модулю от гринвича; `.lon` компонента растёт по модулю от экватора, положительна на севере, отрицательна на юге; таким образом, если компонента `.lat` текущей координаты меньше чем `.lat` левой стороны прямоугольника – находимся вне данной-геозоны; иначе следующая проверка: если `.lon` текущей координаты больше `.lon` верхней стороны прямоугольника – находимся вне геозоны; иначе следующая проверка и т.д. Всего четыре проверки; если все неудачные – значит, находимся внутри геозоны.

4.2.6 Подсистема CTRL

Подсистема `CTRL` не реализует какой-либо API для других подсистем Open AT приложения. Она отвечает за обработку AT команд, поступающих в консоль от пользователя или внешнего процессора.

Для обработки команд `CTRL` выполняет либо обращение к `NAVI`-подсистеме, либо самостоятельную отправку `BINR`-команд и обработку `BINR`-ответов.

Если для получения той или иной информации требуется несколько BINR-ответов, то соответствующие BINR-команды отправляются сразу (т.е. не дожидаясь получения ответа на только что отправленную команду). Обработчики ответов, устанавливая соответствующие флаги, контролируют поступление всей необходимой информации и вызывают конечную функцию выдачи результата в консоль.

Если на момент поступления очередной AT команды предыдущая ещё не обработана, в консоль выводится сообщение вида +<prev_cmd>: BUSY. Таймаут работы каждой AT команды – две секунды. Он контролируется в цикле while (1) потока, по истечению таймаута выдаётся +<cmd>: TIMEOUT.

5 Встраиваемое программное обеспечение MTDS

В качестве альтернативы «Расширению AT команд» компания «3D Телеметрия» предлагает Вам использовать встраиваемое программное обеспечение MTDS. MTDS – это законченное Open AT приложение, реализующее функционал удалённого мониторинга различных объектов, в первую очередь – транспортных средств.

ПО MTDS реализует Журнал событий («чёрный ящик»), поддержку различных внешних датчиков (в том числе, CAN шины), передачу данных на сервер, обновление прошивки по эфиру и другие возможности.

При этом значительная часть ПО MTDS доступна в открытых исходных кодах, что позволит Вам изменять и расширять его функциональность силами собственных инженеров или независимых сторонних компаний.

5.1 Как превратить TE-SL6087-NV08C в полноценный ГЛОНАСС/GPS/GSM трекер за 30 минут

Вы можете прямо сейчас скачать бесплатную версию ПО MTDS и начать использовать его на Вашем модуле TE-SL6087-NV08C. Бесплатная версия ПО MTDS доступна только в бинарном виде (без исходных кодов), и в ней выключена поддержка внешних датчиков. То есть, с помощью этой версии Вы сможете отслеживать только навигационные данные.

Во всём остальном, кроме внешних датчиков, бесплатная версия ПО MTDS полностью функциональна. Установив эту версию на свой модуль TE-SL6087-NV08C, Вы превратите его в полноценный ГЛОНАСС/GPS/GSM трекер совершенно бесплатно. Дополнительная информация – здесь: http://www.3d-telemetry.ru/glonass-gps-gsm/mtds-nv08c-install.html.