Формирование и передача через Интернет

навигационных и служебных сообщений в GPS/3G-модеме MT4100

Статья предназначена, прежде всего, для опытных пользователей продукции Enfora, которые хотят добавить трекеры серии Spider MT в свои системы спутникового мониторинга транспортных средств, но также может представлять интерес для разработчиков различных систем контроля, в которых нужно учитывать пространственное положение и географические координаты.

Виктор Алексеев, к. ф.-м. н.
info@telemetry.spb.ru

Новый модем Novatel MT4100 производства фирмы Enfora объединяет в себе GPS-приемник и высокоскоростной 3G-передающий модуль. Модем оснащен двумя встроенными трехкоординатными акселерометрами и мощным управляющим микроконтроллером. Расширенный интерфейс пользователя содержит цифровые и аналоговые вводы/выводы, два последовательных порта RS-232, контакты для подключения аккумулятора автомобиля и замка зажигания, 1-Wire-интерфейс. Встроенное программное обеспечение (ПО) позволяет конфигурировать передаваемые на центральный сервер сообщения в очень широких диапазонах.

Европейский вариант модема MT4100 называется UMT2200. Он предназначен для работы в диапазоне частот 850/1900 (Bands V, II). Всю навигационную и служебную информацию модем пересылает на сервер в виде UDP/IP- или TCP/IP-сообщений с помощью встроенного высокоскоростного модуля 3G. Подробную информацию о технических параметрах устройства можно найти в [1, 2]. Кроме того, модем может посылать служебные сообщения через UDP-протокол в виде SMS или по e-mail.

Сообщение формируется следующим образом:

- создание статусного сообщения;
- добавление заголовка UDP;
- добавление заголовка IP;
- добавление структуры PPP;
- запись в COM-порт.

С модемом программист может работать в реальном масштабе времени через пользовательский программный интерфейс (Application Program Interface, API) [5].

Модем можно настроить так, чтобы он периодически пересылал на COM-порт только NMEA-сообщения.

Команда выбора формата NMEA-сообщений имеет следующую структуру:

```
AT$GPSLCL=<option>,<nmeaMsgs>.
```

Выбор необходимого NMEA-сообщения производится следующим образом:

```
<nmeaMsgs> =1: GGA;
<nmeaMsgs> =4: - GSA;
<nmeaMsgs> =8: - GSV;
<nmeaMsgs> =16: - RMC;
<nmeaMsgs> =64: - PENFG — служебное сообщение.
```

Параметр <***option***> разрешает (1) или запрещает запись NMEA-сообщений в последовательный порт модема.

Сразу после отработки команды ***$GPSLCL=1,29*** на выходе последовательного порта можно будет наблюдать пакеты четырех NMEA-сообщений:

```
$GPGSV,2,1,08,05,33,057,,13,07,336,,16,34,307,,18,09,1
70,*74
$GPGSV,2,2,08,21,57,206,,25,11,160,,29,56,109,,31,13,2
37,*72
$GPRMC,203802.000,V,5950.209099,N,03022.660528,E,0.0
,0.0,230414,,,N*73
$GPGGA,203802.000,5950.209099,N,03022.660528,E,0,00,
99.0,011.42,M,18.0,M,,*57
$GPGSA,A,1,,,,,,,,,,,,,99.0,99.0,99.0*00
```

Простейший вариант, позволяющий отслеживать на сервере факт перемещения транспортного средства (ТС), — передача только RMC-сообщения в ASCII-формате.

В том случае, когда параметр задан для RMC как ***<nmeaMsgs> =16***, на COM-порт модема с заданной периодичностью (от 1 Гц) будет поступать в реальном масштабе времени строка следующего вида:

```
$GPRMC,203802.000,V,5950.209099,N,03022.660528,E,
0.0,0.0,230414,,,N*73,
```

где:

- 203802.000 — время UTC определения координат;
- V — предупреждение приемника, либо 3 А — данные достоверны;
- 5950.209099,N — широта С/Ю;
- 03022.660528,E — долгота В/З;
- 0.0,0.0 — скорость относительно земли (в узлах);
- дата (ддммгг);
- курс, град. (истинный);
- магнитное склонение, град. В/З;
- индикатор режима 2, 3.

С помощью специальных АТ-команд информация с последовательного порта может быть передана на центральный сервер по беспроводному каналу 3G (GPRS).

Адреса и порты, на которые должно быть отправлено сообщение, задаются командами ***$FRIEND*** и ***$UDPAPI*** для UDP/IP или командой ***$TCPSRC*** для TCP/IP.

Так, например, чтобы сообщение поступило на адрес 212.98.173.108 сервера GURTAM [5] в порт 20122, нужно задать следующие параметры:

```
AT$UDPAPI=,20122
AT$FRIEND=01,1,"212.098.173.108",20122,3
```

В зависимости от заданных пользователем дополнительных параметров сообщение может быть в дальнейшем передано на сервер и сохранено/удалено из памяти модема; сохранено в памяти модема и отправлено на сервер по дополнительному запросу либо по заданному алгоритму.

При передаче сообщения через Интернет перед информационными данными записывается заголовок в виде: ***<00><05><02><00>***, где ***<00><05>*** — коды API; ***<02>*** — API командная строка; ***<00>*** — длина дополнительного расширенного заголовка (в данном примере его нет).

Полученное RMC-сообщение достаточно просто привязать к карте местности. Проще всего это сделать с помощью Google API или GPS Babel.

Алгоритм управления модемами серии Spider MT использует идею «обработки событий» — Novatel Wireless Event Engine. Модем может быть запрограммирован таким образом, чтобы внешние события отслеживались и вызывали ответные действия определенного рода.

В качестве внешних событий могут быть рассмотрены, например, таймер, превышение скорости движения, резкое ускорение или торможение, перегрев двигателя и другие подобные случаи. В качестве ответного действия модем может посылать на заранее заданные адреса сложные информационные сообщения, а также запускать таймер и счетчик событий, устанавливать время аварийного будильника, изменять параметры пользовательских вводов/выводов и др. Алгоритм команды охватывает практически все стандартные ситуации при работе с M2M-приложениями.

Структура команды, описывающей события, выглядит следующим образом:

```
AT$EVENT=<Event group>,<Response>,<Category>,
<Param1>,<Param2>,<Param3>.
```

<Event group> определяет группу событий. Все события могут быть дополнительно разбиты по отдельным группам, которые будут обрабатываться поочередно. Кроме единичных событий, можно задать также и множественные, последовательные события, такие как «Состояние вводов/выводов», «Сетевые IP-события» и т. д. Множественные события внутри одной группы должны рассматриваться как логическое условие «И».

<Event Response type> определяет тип события: «Входное» (Input) или «Выходное» (Output). Этот параметр может принимать значения «0», «1» (входное событие) или «2», «3» (выходное событие).

<Event category> описывает конкретные входные и выходные события.

ПО модема MT4100 в настоящее время насчитывает 214 входных событий и 154 выходных. Все они подробно описаны в [3]

<Param1> задает диапазон входных событий и тип выходных событий.

<Param2> определяет битовую маску сообщения.

<Param3> используется только совместно с событиями 40, 41, 42 для отправки на сервер UDP-сообщения и с событием 52 для отправки на сервер TCP-сообщения.

Новые модемы серии MT поступают с завода-изготовителя с минимальным активным набором команд, описывающих только события регистрации в сети:

- ***AT$EVENT=1,0,9,2,4*** (1-я группа событий; входное событие № 9 — автоматическая регистрация в сети GSM);
- ***AT$EVENT=1,3,39,1,0,0*** (1-я группа событий; выходное событие № 39, идет регистрация, включение индикаторного диода «GSM»);
- ***AT$EVENT=2,0,9,5,5*** (2-я группа событий; входное событие № 9 — автоматическая регистрация в сети GSM);
- ***AT$EVENT=2,3,39,4,0,0*** (2-я группа событий; выходное событие № 39, идет регистрация, диод «GSM» включается с частой 1 Гц);
- ***AT$EVENT=3,0,9,0,0*** (3-я группа событий; входное событие № 9 — нет регистрации в сети GSM);
- ***AT$EVENT=3,3,15,0,0,0*** (3-я группа событий; выходное событие № 15, установка ввода/вывода GPIO #8 в качестве выхода с низким уровнем Low «0»);
- ***AT$EVENT=4,0,9,1,1*** (4-я группа событий; входное событие № 9 — успешная регистрации в сети GSM);
- ***AT$EVENT=4,3,39,4,0,0*** (4-я группа событий; выходное событие № 39, идет регистрация, диод «GSM» горит постоянно).

Модем может передавать на центральный сервер по каналу 3G как навигационные данные, получаемые GPS-модулем, так и дополнительную информацию от акселерометров, АЦП, однопроводных датчиков. Кроме того, сообщение может содержать дополнительную служебную информацию. Модем может посылать на сервер UDP-, TCP-, USSD-сообщения, а также SMS- и e-mail-сообщения на заданные адреса и номера телефонов.

В памяти модема записана программа (script), содержащая дополнительные входные и выходные события. Эта программа не активирована по умолчанию. Поэтому новый модем умеет только самостоятельно регистрироваться в сети и посылать об этом сообщение на сервер.

Проверить, какой именно скрипт зашит, но не активирован в вашем модеме, можно с помощью команды:

```
AT$FFS=3
at$ffs=3
```

Ответ, например, будет такой:

```
$FFS: script1.txt
```

Используя команду ***AT$ATEXEC=<FILENAME>,<OPTION>,*** можно активировать нужный скрипт.

<FILENAME> — имя файла, содержащего скрипт, например *script1.txt*;

<OPTION> =0 — скрипт отрабатывается полностью, даже в тех случаях, когда одна или несколько команд посылают сообщение об ошибке;

<OPTION> =1 — скрипт отрабатывается до первой ошибки;

<OPTION> =2 — скрипт отрабатывается до первой ошибки, при этом модем автоматически перезагружается.

Команда активации зашитого в памяти модема скрипта ***$ATEXEC*** может быть передана через кабель на COM-порт, по сети Интернет, а также посредством SMS. Проверка действия команды ***$ATEXEC*** осуществляется следующим образом:

```
AT$ATEXEC?
$ATEXEC:<script1.txt>,<0>.
```

Команды и события, записанные в активированном скрипте, можно посмотреть с помощью команды ***AT&V***. Текст команд анализируется и редактируется с помощью прикладного ПО Enfora Toolbox.

На рис. 1 показаны два интерфейсных окна блока **AT&V Parcer** программы Toolbox. В левом окне вводятся результаты команды ***AT&V***. В правом окне выводятся все события, соответствующие введенным данным ***AT&V***.

В правом интерфейсе можно также открывать другие вкладки с параметрами скрипта:

- **Timers** — установки всех таймеров;
- **Stored Events** — сохраненные результирующие события (команда ***$STOATEV***);
- **Geo/Poly fence** — заданные гео- и полизоны;
- **Modem setup** — первоначальные базовые установки модема;
- **All** — полностью весь скрипт (пример для этого варианта показан на рис. 2).

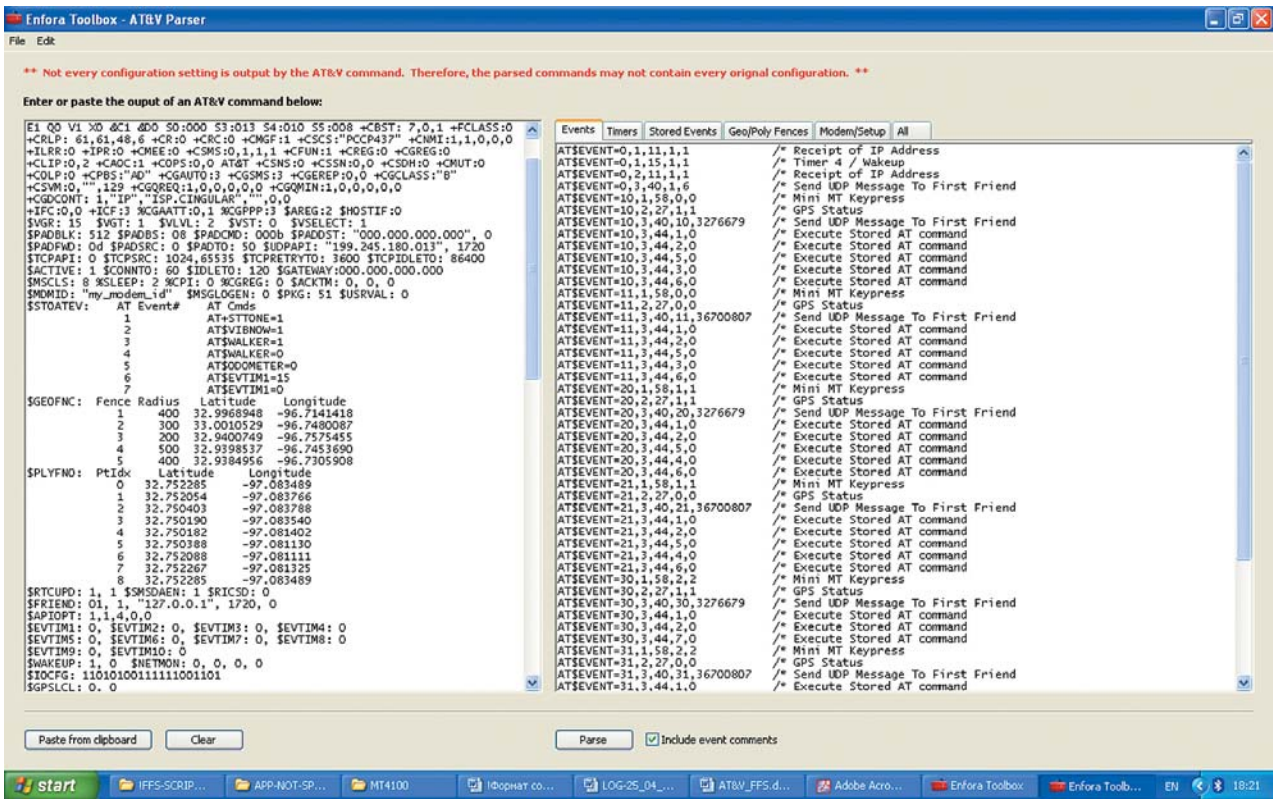

Рис. 1. Интерфейсы блока **AT&V Parcer** программы Enfora Toolbox

В программном обеспечении Enfora есть понятия «гео-зона» (Geo Fencing a Circular Area) и «поли-зона» (polygonal area).

Команда ***$GEOFNC*** позволяет получать сообщение в том случае, когда модем пересек определенную круговую географическую область, которая определяется центром с заданными координатами и радиусом. Команда ***AT$PLYFN#*** дает возможность определить до 25 отдельных полигональных географических зон (# может быть любым числом из диапазона 0–24). При пересечении границ заданного многоугольника на сервер будет отправлено соответствующее сообщение. Каждая из команд определяет одну из возможных вершин и углов многоугольника. Отрезки многоугольника создаются путем последовательного подключения ненулевой вершины к предыдущей (например, точка 0 подключена к точке 1, которая подключена к точке 2, и т. д.). Полигон может быть определен в любом направлении по часовой стрелке или против.

Редактировать скрипт, добавлять в него новые команды и события можно с помощью интерфейса **Script Generator** программы Enfora Toolbox (рис. 3).

В левом окне программы выбирается необходимая модель и сценарий группы событий. Например,

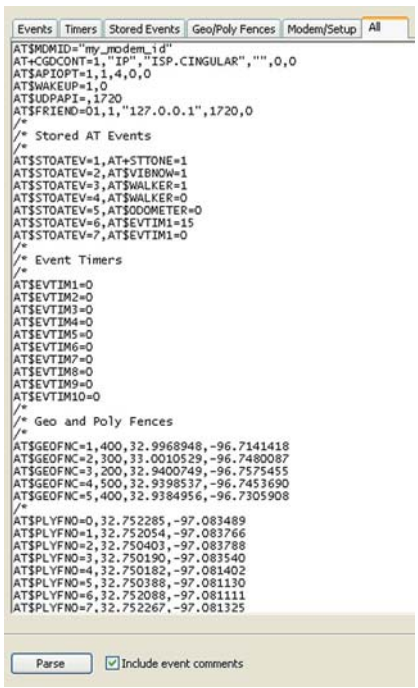

Рис. 2. Окно полного описания скрипта блока **AT&V Parcer** программы Enfora Toolbox

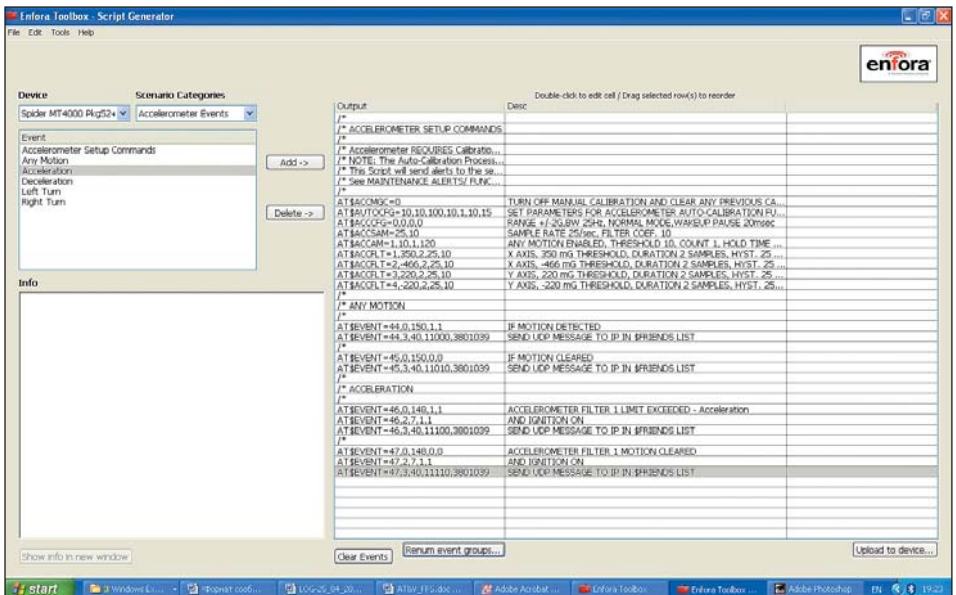

Рис. 3. Окно **Script Generator** программы Enfora Toolbox

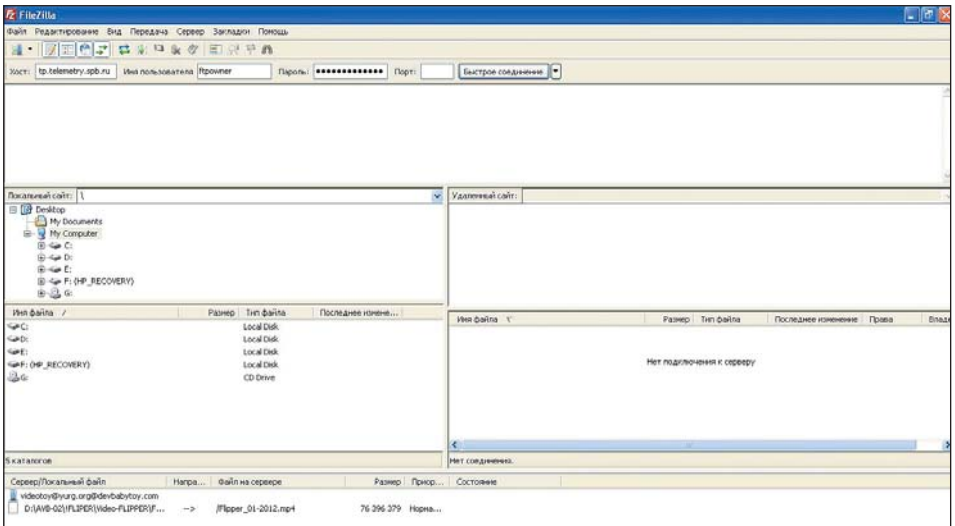

Рис. 4. Главное окно программы FTP-менеджера FileZilla

MT4000 и Accelerometer Events. Также в этом окне нужно выбрать нужную группу событий, например «Ускорение» или «Торможение». При нажатии на кнопку **Add** нужная группа событий будет добавлена в скрипт, показанный в правом окне. Нумерации групп событий можно изменить в соответствии с другими блоками программы с использованием кнопки **Renum Event Groupe**.

Окончательный вариант скрипта сохраняется в виде файла на компьютере: **File–>Save with comment–>Save without comment**.

Опция **Upoad to device** дает возможность загрузить готовый скрипт в модем по кабелю через RS-232. Также в модем можно загрузить скрипт, сохраненный в виде файла, по Интернету через FTP-сервер. Для закачки исполнительного файла в удаленный модем используется набор специальных AT-команд. Подробно об этом будет сказано далее.

Передача файла через Интернет проводится по протоколу FTP с использованием TCP/IP. В самом протоколе поддерживаются средства для докачки файла для тех случаев, когда передача была прервана по каким-либо причинам. При этом пользователи, для которых предназначены материалы, смогут использовать для скачивания файлов стандартные FTP-клиенты, обеспечивающие докачку и скачивание в несколько потоков.

Протокол FTP реализуется по схеме «клиент–сервер». При этом используются разные сетевые соединения для передачи команд и данных между клиентом и сервером. Так, например, команды и данные передаются на разные порты (порт 20 — данные, порт 21 — команды). Существует несколько вариантов аутентификации. Один допускает передачу логина и пароля открытым текстом. В другом случае пользователь может подключиться к серверу анонимно. Кроме того, есть варианты, позволяющие использовать протокол SSH для безопасной передачи, при которой логин и пароль зашифрованы и скрыты.

Можно, например, использовать бесплатный FTP-сервер FileZilla. Загрузить его бесплатную версию можно с сайта проекта [6]. Следует учитывать тот факт, что если IP-адрес центрального ПК статический, то никаких проблем не возникнет. В случае, если адрес динамический, при каждом подключении центрального ПК к Интернету IP будет другим, и нужно сообщать пользователям этот новый адрес FTP-сервера. В этом случае можно использовать любой сервис Dynamic DNS, на котором центральный ПК может зарегистрировать свое доменное имя.

После включения компьютера и запуска программы FTP-сервер будет находиться в режиме ожидания запросов от модема (рис. 4). Для продолжения работы нужно ввести пароль (рис. 5). Подробная инструкция по работе с FTP-менеджером FileZilla приведена на сайте [7]. Следует подчеркнуть, что для надежной работы сервер должен поддерживать работу с AT-командами и незапрашиваемыми откликами, используя формат Enfora UDP API messages.

Модем (FTP-клиент) первоначально должен быть сконфигурирован для работы с FTP-сервером с помощью AT-команды:

```
AT$FOTACFG="ftpServer",,»username»,"password",,,,0.
```

В этой команде определяется, кроме прочего, режим загрузки — автоматический или ручной (последний параметр «0» или «1»).

С помощью AT-команд можно открыть сессию с удаленным модемом по порту 21. Это соединение будет оставаться открытым на время работы сессии. Второе соединение, необходимое для передачи данных, может быть инициализировано как сервером из порта 20 к порту соответствующего модема (активный режим), так и удаленным модемом из любого порта к порту сервера (пассивный режим).

По команде ***AT$FOTAGET="remotefilename"*** модем инициализирует соединение с FTP-сервером через IP-протокол и создает контрольное

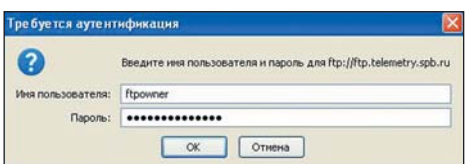

Рис. 5. Запрос на аутентификацию для работы с FTP-менеджером FileZilla

соединение. Поскольку протокол FTP работает только через TCP-соединение и последовательный порт, то передача данных происходит в два этапа. На втором этапе данные в виде IP-пакета пересылаются в буфер временной памяти модема. Затем они конвертируются в поток последовательных данных, пригодный для передачи через COM-порт.

Файл со скриптом, который нужно удаленно загрузить в модем, может быть записан в папке *MT4100*, находящейся в корневом каталоге, например *C:\Enfora\MT4100*. Важно то, что этот файл должен храниться в той директории на FTP-сервере, к которой модем может получить свободный доступ.

Следующий шаг заключается в том, что нужно загрузить прошивку в программу FTP-менеджера. Для этого нужно с помощью указателя мыши «перетащить» файл из левого окна интерфейса, в котором показаны каталоги клиента на жестком диске, в правое окно, соответствующее месту на FTP-сервере.

Пользователь может контролировать процесс получения модемом файла данных с помощью AT-команды ***AT$FOTAGET?***. Если процесс обновления ПО прошел успешно, ответ на эту команду будет следующий:

```
$FOTAGET: 0, fota.bin, 0, 0,0.
```

Со своей стороны, сервер обеспечивает контроль передачи данных через последовательный порт. После окончания получения данных модем посылает уведомление об успешной передаче. Подробно процесс записи скриптов в память модема описан в [8]. Сообщение может быть послано в бинарном или ASCII формате. Структура сообщений формируется битовой маской (bit mask), которую пользователь задает по своему усмотрению, в зависимости от конкретной задачи. Битовая маска соответствует параметру ***<Param2>*** в команде ***AT$EVENT***. Поэтому структура сообщения, обусловленная тем или иным событием, будет меняться в зависимости от типа и категории события.

В таблице 1 показаны основные варианты различных видов битовых масок, обуславливающих структуру сообщения, которое модем MT4100 отправляет на сервер, в зависимости от выходного события. Приведены максимально возможные наборы передаваемых параметров. Естественно, количество параметров можно сократить до необходимого минимума.

В третьем столбце таблицы приведен пример, когда в байтах 3, 4, 5 передается информация, которая раньше была сохранена в специальных таблицах (User Var Table). В программном обеспечении Enfora есть утилита **User variables** («Переменные пользователя»), которая дает возможность сохранять с дополнительным расширением значения выбранных параметров во flash-памяти модема. Эта функция позволяет отправлять разные формы отчетов в одном сообщении. В альтернативном варианте нужно было бы для каждой формы отчета использовать разные события и битовые маски и отправлять каждый раз отдельно разные сообщения [9]. Значения User Var Tab автоматически сохраняются при сбросе ПО. Как правило, **User variables** используется совместно с функцией

Таблица 1. Базовые варианты битовых масок для различных видов событий

Код битовой маски	Основные параметры (Bitmask 13421772)	Дополнительные параметры: акселерометр, iButton (Bitmask 2179727359)	Дополнительные параметры: OBDII (Bitmask 947912647)	Дополнительные параметры: GARMIN (Bitmask 2147483647)
Bit0	Send Binary message	Send Binary Message	Send Binary Message	Send Binary Message
Bit1	Param 1 (4 bytes)	Param 1 (4 bytes)	Param 1 (4 bytes)	Param 1 (4 bytes)
Bit2	Modem ID (22 bytes)	Modem ID (22 bytes)	Modem ID (22 bytes)	Modem ID (22 bytes)
Bit3	GPIO (2 bytes)	User Var Bitmask – Bit 0		Garmin Connection Status (1 bytes)
Bit4	Analog/Digital 1 (2 bytes)	User Var Bitmask – Bit 1		Garmin Product ID (8 bytes)
Bit5	Analog/Digital 2 (2 bytes)	User Var Bitmask – Bit 2		Garmin V1 Txt Msg Ack Info (27 bytes)
Bit6	Store Msg If No GPRS Coverage	Store msg if no GPRS coverage	Store msg if no GPRS coverage	Store Msg If No GPRS Coverage
Bit7	Input Event Number (1 bytes)	Input Event Number (1 bytes)	Input Event Number (1 bytes)	Input Event Number (1 bytes)
Bit8	GPS Date (3 bytes)	Accelerometer XYZ Running Avg (6 bytes)	GPS Date (3 bytes)	Garmin Open TXT Msg Info (92 bytes)
Bit9	GPS Status (1 bytes)	Accelerometer XYZ Filter X1 Values (6 bytes)	GPS Status (1 bytes)	Garmin Stop Status (8 bytes)
Bit10	GPS Latitude (3 bytes)	Accelerometer XYZ Filter X2 Values (6 bytes)	GPS Latitude (3 bytes)	Garmin ETA Status (24 bytes)
Bit11	GPS Longitude (4 bytes)	Accelerometer XYZ Filter Y1 Values (6 bytes)	GPS Longitude (4 bytes)	Garmin Date (4 bytes)
Bit12	GPS Speed (2 bytes)	Accelerometer XYZ Filter Y2 Values (6 bytes)	GPS Speed (2 bytes)	Garmin Time (4 bytes)
Bit13	GPS Heading (2 bytes)	Accelerometer XYZ Filter Z1 Values (6 bytes)	GPS Heading (2 bytes)	Garmin Latitude (4 bytes)
Bit14	GPS Time (3 bytes)	Accelerometer XYZ Filter Z2 Values (6 bytes)	GPS Time (3 bytes)	Garmin Longitude (4 bytes)
Bit15	GPS Altitude (3 bytes)	GPS Data Bitmask – Bit 0	GPS Altitude (3 bytes)	Garmin Altitude (4 bytes)
Bit16	GPS Number Of Satelites (1 bytes)	GPS Data Bitmask – Bit 1	GPS Number Of Satelites (1 bytes)	Garmin Speed (4 bytes)
Bit17	Disable Messages In Low Power Mode	GPS Data Bitmask – Bit 2	Disable Messages In Low Power Mode	Garmin PVT Fix Type (2 bytes)
Bit18	Send SMS When GPRS Not Available		Send SMS When GPRS Not Available	Garmin A604 Open Txt Msg Info (24 bytes)
Bit19	Last Known GPS Data When GPS Status Not Available	iButton Driver ID (8 bytes)	Last Known GPS Data When GPS Status Not Available	Garmin Canned Response Refresh List (25 bytes)
Bit20	GPS Odometer (4 bytes)		GPS Odometer (4 bytes)	GPS Trip Odometer (4 bytes)
Bit21	RTC Time (6 bytes)	RTC time (6 bytes)		RTC Time (6 bytes)
Bit22	Short Modem ID (8 bytes)	Short modem ID (8 bytes)	Short Modem ID (8 bytes)	Short Modem ID (8 bytes)
Bit23	Battery Level (Mini-MT Only) (1 bytes)	Excessive Acceleration Data (5 bytes)		Garmin Update Canned Msg List Flag (1 bytes)
Bit24	GPS Overspeed Data (6 bytes)	Excessive Deceleration Data (5 bytes)		Garmin Msg Status (16 bytes)
Bit25	PCELL Data 0-5 (56 bytes)			Garmin Driver ID (49 bytes)
Bit26	GPS Alternate Overspeed (6 bytes)			Garmin Driver Status List Flag (1 bytes)
Bit27			OBDII Data VIN/Protocol/PKG/RSSI (26 bytes)	Garmin Driver Status (4 bytes)
Bit28			OBDII MIL Data (25 bytes)	Garmin Ping (4 bytes)
Bit29			OBDII trip Odometer (4 bytes)	Garmin Throttle List Status (Variable) (242 bytes)
Bit30				Bitmask Table ID – Low Order Bit
Bit31		Bitmask Table ID – High Order Bit		

Event Processing в качестве расширения «входных триггерных событий». Такой алгоритм дает пользователю возможность передавать данные, которые невозможно включать в стандартный выходной формат, зафиксированный параметром <***Param2***> определенного события. Стандартный вариант показан во второй колонке таблицы 1. Варианты с использованием **User variables** показаны в колонках 3, 4, 5.

Функция **User variables** используется совместно с выходным событием № 128.

Например, команда ***AT$EVENT=99,3,128,3,9*** копирует результат входного события № 9 (регистрация в сети GSM) в стек переменных пользователя № 3 (**User Variable 3**).

При использовании функции **User variables** можно задавать значение таймера таким образом, чтобы сообщение всегда сохранялось в стеке переменных пользователя, даже в случае полного аппаратного сброса модема (например, потеря питания).

Пример данных в двоичном формате, которые модем передает на сервер, выглядит так:

```
0x000502000000000A00000C022543015031 86FFFF6F84000000000019353000038080B060
E0A2033020302040001050082 0A58004F150000820000004506002B1A9004D0E00820A5000
51130020000000550E0000000000000000000000000000
```

В этом сообщении содержится следующая информация:

- байты 1–2 (0005) — необработанные коды параметров интерфейса пользователя (API parameters) [4];
- байт 3 (02) — командный;
- байт 4 (00) — длина заголовка API (API Optional Header). Длина «0» означает, что в сообщении нет API Optional Header;
- байты 5–35 (0000000A00000C022543015031 86FFFF6F84000000000019353000038080B060 E0A2033) — данные (описание приведено в таблице 2);
- байты 36–92 (02030400010500820A58004F1 5000082000000450600 2B1A9004D0E00820A 500051130020000000550E0000000000000000 000000000000) — параметры сети сотовой связи, PCELL-данные (Mobile Country Code,

Таблица 2. Значения параметров в строке данных для сообщения, отправленного на сервер модемом MT4100

Значение параметра (формат Hex)	Значение параметра (формат ASCII)	Описание параметра
00 00 00 0A	10	Param1 – параметр в команде выходного события
00 00	0	A/D2 – показания АЦП2
0C	12	Input Event Number 12 – номер входного события 12 (срабатывание таймера 1)
02 25 43	140611	GPS Date – дата (ддммгг)
1	1	GPS Status – статус сигнала спутников
50 31 86	5255558	GPS Lat – широта (52.926N)
FF FF 6F 84	-36988	GPS Long – долгота (0.616W)
00 00	0	GPS Speed – скорость (knots × 10)
00 00	0	GPS Heading – направление вектора скорости
01 93 53	103251	GPS Time – GPS-время (ч:мин:с)
00 00 38	56	GPS Altitude – высота в метрах
8	8	Number Of Satellites – количество видимых спутников
0B 06 0E 0A 20 33	11 06 14 10 32 51	RTC Time – мировое время (гг мм дд ч мин с)

Mobile Network Code Location Area Code, Cell Identifier и др.).

Команда ***AT$APIOPT*** добавляет в различных вариантах API Optional Header в заголовок UDP API- и TCP API-сообщения.

Структура этой команды ***AT$APIOPT=<MDMID>,<Msg Event Format>,<Event SeqNum>,<HdrDisable>,<Output Event Type>,<HexModemID>,<SendParam3>*** позволяет редактировать в широких пределах заголовки UDP API- и TCP API-сообщений.

Например, при значениях параметров ***$APIOPT: 1,1,4,0,1,0,0,*** в сообщение можно включать такие дополнительные заголовки, как:

- ***<MDMID>*** — идентификационный номер MDMID;
- ***<Msg Event Format>*** — формат сообщения о выполненных событиях (битовая маска ***<Param2>***);
- ***<Event SeqNum>*** — заголовок будет содержать все четыре байта последовательности выполненных событий, без ограничений;
- ***<HdrDisable>*** — расширенный заголовок для всех вариантов сообщений;
- ***<Output Event Type>*** — номер выходного события;
- ***<HexModemID>*** — идентификационный номер IMEI;
- ***<SendParam3>*** — нет.

Соответствующее этим настройкам сообщение содержит расширенный заголовок API Optional Header и выглядит иначе, чем сообщение, рассмотренное в предыдущем примере:

```
0005022E11013031333739393030303030363532300602003
9FFC70603000002A503062808070C8CD47E5F78060900000
000000003E8202020202020303133373939303030303036 35
3230200803F8DA095ACD45002E207F000005F102589F0000
250800000000000101000003
```

Здесь байты 1–2 (0005) — необработанные коды параметров интерфейса пользователя (API parameters, Binary event data) [4]; байт 3 (02) — командная строка API (general status information); байт 4 (2E) — длина заголовка API (46 байтов); байты 5–50 — заголовок API, включающий:

- имя модема, заданное пользователем (OPT-HDR_MODEM) 11013031333739393030303030 3036353230;
- битовая маска 3801031 (OPT-HDR_FORMAT_CODE_PARAM2) 06020039FFC7;
- код последовательности выполненных событий (OPT-HDR_SEQ_NBR) 0603000002A5;
- номер выходного события «40» — отправка UDP на заданные адреса (OPT-HDR_OUTPUT_EVENT) — 030628;
- идентификационный номер в формате HEX (OPT-HDR_HEX_MODEM) 08070C8CD47E5F78.

В остальных байтах передаются данные в соответствии с заданной битовой маской:

- идентификационный номер 2020202020203 03133373939303030303036353230 20;
- входное событие «8» — регистрация в сети GSM — 08;
- навигационные данные GPS (координаты, скорость, время, высота, направление, одометр, количество спутников, RTC) 03 F8 DA 09 5A CD 45 00 2E 20 7F 00 00 05 F1 02 58 9F 00 00 25 08 00 00 00 00 00 01 01 00 00 03.

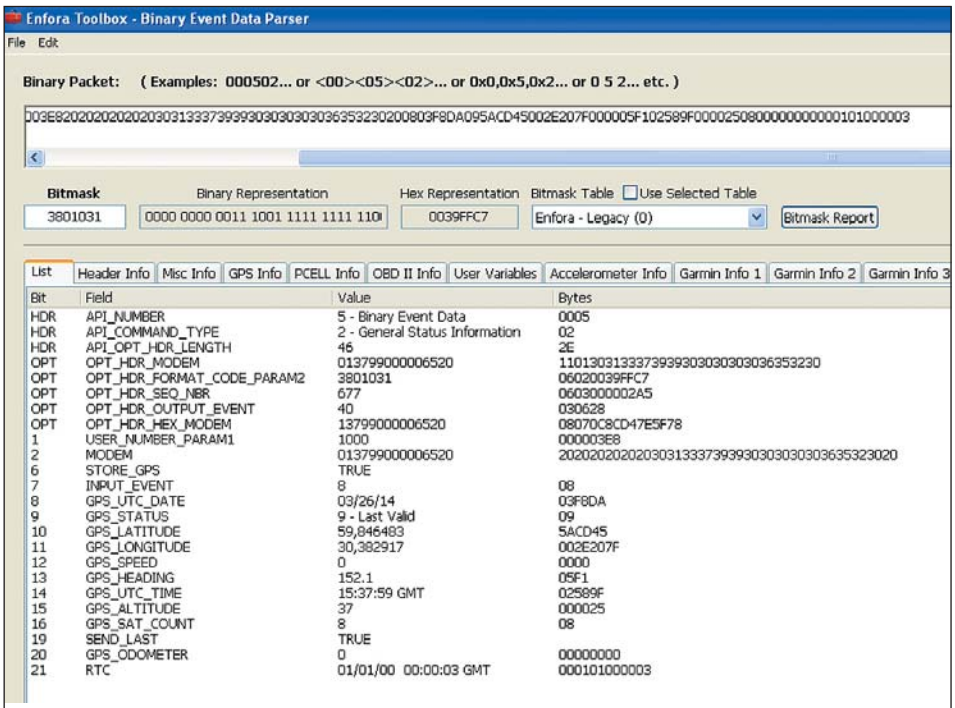

Рис. 6. UDP-сообщение, которое модем MT4100 отсылает на сервер после подачи питания, декодированное с помощью программы Enfora Toolbox

Для расшифровки бинарных сообщений удобно использовать раздел **Binary Event Data Parcer** в программе Enfora Toolbox. На рис. 6 показано декодированное с помощью этой программы первое сообщение, которое модем отсылает на сервер после подачи питания.

Модем начинает отсылать сообщения на сервер сразу после подачи напряжения. Первые сообщения, как правило, формируются на основе служебных событий:

- Event 8 — питание на модем подано;
- Event 9 — регистрация в сети GSM;
- Event 10 — установление связи 3G (EDGE, GPRS);
- Event 11 — получение IP-адреса.

Разные информационные сообщения целесообразно передавать в разных пакетах. Например, если мы хотим получать геодезические данные каждые 10 с, то нужно отправлять их отдельно, в коротких сообщениях без дополнительного заголовка. Для этого нужно сначала убрать дополнительный заголовок командой ***AT$APIOPT=0,0,0,1,0,0,0.***

Целесообразно в этом случае воспользоваться функцией **User variables** и выходным событием № 128, которое копирует результат соответствующего входного события в стек переменных пользователя. Далее нужно разрешить передачу в последовательный порт модема навигационных данных, которые принимает GPS:

```
AT$GPSLCL=1,29
```

Чтобы отправлять сообщение с координатами каждые 10 с, нужно задать таймер и создать соответствующее входное событие:

```
AT$EVTIM1=10
```

AT$EVENT=16,1,12,1,1 (входное событие № 12 в группе №16, означающее тот факт, что сработал таймер № 1).

Следует обратить внимание на нумерацию группы событий. ПО Enfora Toolbox позволяет переименовывать заданные события в любой последовательности.

Затем нужно рассчитать битовую маску для передачи минимально необходимого для решения данной конкретной задачи количества информации. Для этой цели можно воспользоваться утилитой **Bitmask calculator** в прикладном ПО Enfora Toolbox. Нет никакого смысла ставить «1» в каждом бите соответствующего события. Нужно выбрать оптимальный вариант.

Результаты вычислений для этого примера показаны на рис. 7.

Итоговая команда в данном случае запишется так:

```
AT$EVENT=16,3,40,1000, 4063175
```

В результате этой команды на сервер, определенный командами ***$FRIEND=01*** и ***$UDPAPI***, каждые 10 с будет отправляться короткое (47 бит) UDP-сообщение, содержащее GPS-значения (координаты, скорость, время, направление движения, количество видимых спутников):

```
000502000000277420202020202030313337393930303030 3
03635323020 0C046E0A015AC82A002E1F3E01B2062C00EB
410000130E00000F050E031D060218
```

Декодированные с помощью программы Toolbox данные показаны на рис. 8.

Другие данные, получаемые модемом от внешних и встроенных блоков, можно отправлять в дополнительных сообщениях

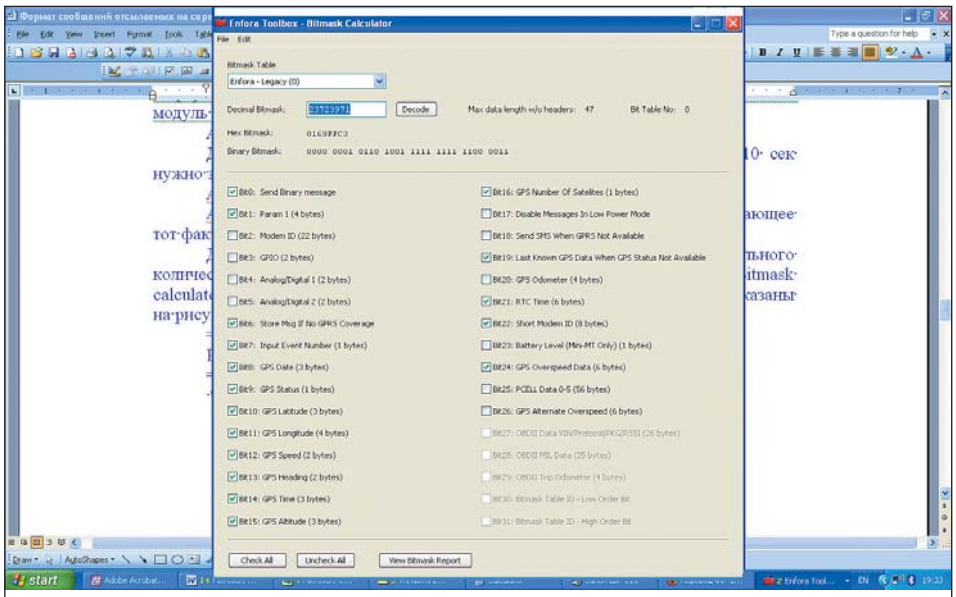

Рис. 7. Вычисление битовой маски с помощью ПО Enfora Toolbox

со своими конкретными битовыми масками. Так, например, температурные данные внешних датчиков модем способен передавать только в бинарном формате в строке данных, в форме двухбитового целого числа со знаком плюс или минус. Следует обратить внимание на то, что в битовой маске выходных событий для датчиков T1 и T2 обязательно должны быть параметры <***Param3***>, равные соответственно 4 и 8.

Чтобы задать <***Param3***> в заголовке сообщения, нужно воспользоваться командой ***AT$APIOPT***. Например, мы зададим параметры заголовка следующим образом:

```
AT$APIOPT=1,1,4,0,0,1
```

В данном примере будет разрешена передача сообщения с HFCIBHTYYSV заголовком UDP API header, включающим в себя следующие параметры:

- идентификационный номер MDMID;
- формат событий;
- номер последовательности событий;
- тип выходного события;
- <***Param3***>.

Имеет смысл сохранить данные в стеке пользователя с помощью **User variables**.

Текущие температурные данные могут передаваться наряду с другими данными через разные интервалы времени. Для этой цели используется стандартное выходное событие, аналогичное тому, которое было использовано выше для получения навигационных данных.

AT$EVTIM7=60 (установка таймера № 7 на 60 с);

AT$EVENT=30,1,68,1,1 (срабатывание таймера № 7);

AT$EVENT=30,3,42,50, 3014655,12 (отправка UDP-сообщения на заданный IP-адрес сервера, определенный командами ***$FRIEND=02*** и ***$UDPAPI***).

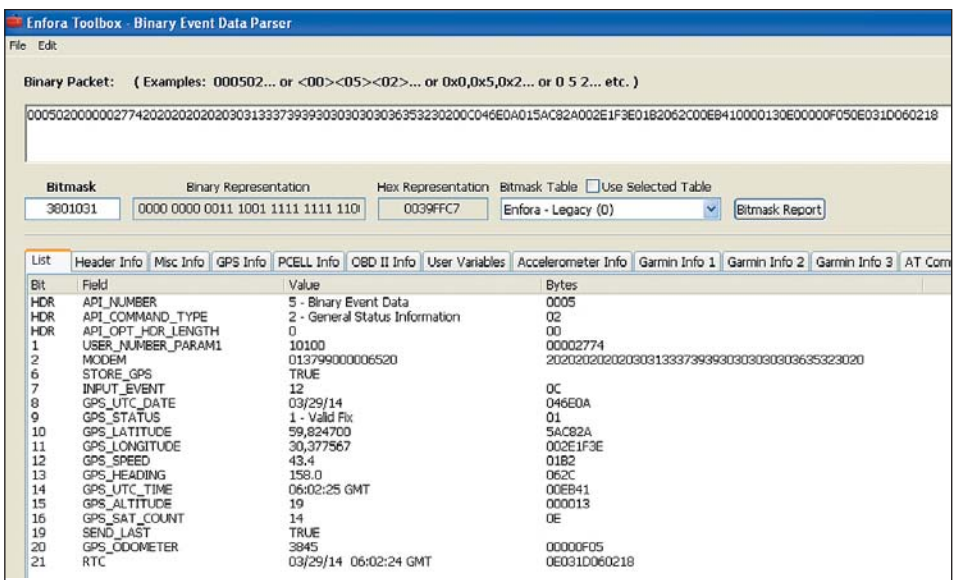

Рис. 8. Окно программы Toolbox с навигационными данными, которые модем MT4100 посылает на сервер каждые 10 с

Этот пример отличается от приведенного выше скрипта, описывающего отправку навигационных данных каждые 10 с, тем, что в последнем случае использован другой набор параметров <***Param1***>,<***Param2***>,<***Param3***>. Следовательно, и содержание UDP-сообщений в этих двух случаях будет разное.

После того как мы ввели эту новую группу событий, навигационные данные попрежнему будут передаваться каждые 10 с. Одновременно с этим каждые 60 с будут передаваться дополнительные данные, поступающие от однопроводных температурных датчиков.

Ниже приведен пример сообщения, содержащего данные с температурой однопроводных датчиков, которое модем MT41000 передает с интервалом 60 с на центральный сервер:

```
000502180601544540500602002DFFFF060300000C880609
0000000C00000032202020202020202020202020202020202
054454D5020C016000000000C0446FA014E1B2A002E202C
00000BC301EAD000001112000000000000300DA03FFAE
```

В этом сообщении, байты 1–3 (000502) — коды API; байт 4 (18) — 24 байта, длина заголовка. Байты 5–23 (060154454D500602002 DFFFF060300000C88) соответствуют строкам расширенного заголовка API, заданным параметрами команды ***$APIOPT=<MDMID>, <Msg Event Format>,<Event SeqNum>,<HdrDisable>, <Output Event Type>,<HexModemID>.*** Байты 24–29 (06090000000C) содержат пакет дополнительной информации о параметре <***Param3***>. В этом пакете: 06 — количество байтов в описании <***Param3***>; 09 — поле заголовка для хранения <***Param3***> в соответствии со структурой API; 0000000C — значение <***Param3***> (T1+T2 = 4+8). Далее идут байты, соответствующие битовой маске, показанной в первом столбце таблицы 1 (Bit 1 – 16, 18, 19, 21). Например, байты с 30 по 33 (00000032) соответствуют <***Param1***>, равному 50. Последние шесть байтов сообщения (0300DA03FFAE) — информация об однопроводных датчиках, подключенных к модему MT4100 по схеме «звезда». Эту строку можно декодировать следующим образом: 0300DA03FFAE — температурные данные, где 03 — количество байтов с результатами первого температурного датчика (три); 00DA — измеренное значение температуры (в десятичном формате это значение равно 218, что соответствует температуре +21,8 °C); 03 — количество байтов с результатами второго температурного датчика (три); FFAE — измеренное значение температуры (в десятичном формате это значение равно –82, что соответствует температуре –8,2 °C).

Следует также отметить, что для управления однопроводными датчиками T1 и T2 в ПО MT4100 введены дополнительные входные события Input Event 212 и Input Event 213. Эти события позволяют работать с аварийными, пороговыми температурными событиями. (Выше были рассмотрены события, связанные с текущими измерен-

ными данными, которые температурные датчики регулярно выдают на COM-порт модема). Пороговые значения температур характеризуют аварийную ситуацию, при возникновении которой диспетчер должен принимать ответные меры.

Пороговые значения для внешних температурных датчиков задаются с помощью команды ***AT$OWCFG=<dev_enable>,<Min Temp>,<Max Temp>,<Time>,*** где:

- ***<dev_enable>*** — включить («1») или выключить («0») температурный датчик T1;
- ***<Min Temp>*** — нижний порог температуры;
- ***<Max Temp>*** — верхний порог температуры;
- ***<Time>*** — время действия события.

Например, границы аварийного события, соответствующего попаданию температуры T1 в интервал 1–10 °C, задаются пороговой командой ***AT$OWCFG1=1,1,10, 60.*** Если температура остается в заданном интервале более 60 с, то наступает аварийное входное событие ***AT$EVENT=13,1,212,1,1.*** Для выходного события можно выбрать передачу на сервер или телефон соответствующего сообщения или дистанционное включение/выключение нагревателя.

В качестве другого примера использования однопроводного интерфейса можно привести кнопку iButton для идентификации водителя. Каждая такая кнопка имеет свой уникальный 64-битовый идентификационный номер, например 2700000095C33108.

Структура номера кнопки iButton: первые 8 бит — идентификационный ключ серии кнопки; следующие 48 бит — уникальный серийный номер; последние 8 бит — CRC-код (дополнительный контрольный код проверки первых 56 бит). Принцип работы кнопки iButton хорошо известен. При соприкасании контактов «таблетки» с приемным устройством идентификационный номер считывается с частотой до 10 раз в секунду. Каждый раз, когда считывается правильный код, его значение сообщается модему через входное событие Event 55:

```
AT$EVENT=,55,<Param1>,<Param2>.
```

Параметры могут принимать значения «0», «1» или «2», которые означают, что считанное значение совпадает («2»), либо не совпадает («1») со значением, ранее сохраненным в устройстве. Значение «0» означает, что номер кнопки не записан в памяти модема. Подробно этот вопрос рассмотрен в [3].

Поскольку утилита **Event Engine** циклически отрабатывается четыре раза в секунду, данное входное событие Event 55 может быть зарегистрировано модемом несколько раз в течение интервала времени, пока устройство iButton находится в контакте со считывателем. Добавить информацию iButton в очередное UDP-сообщение можно двумя способами. В одном случае можно выбрать битовую маску с помощью дополнительной таблицы Bit-Field Table 2–(1,0), разработанной специально для серии MT4000. В этой таблице 19-й бит разрешает («1») или не разрешает передачу информации iButton в сообщение.

Структура данных этой части сообщения показана на рис. 9.

В другом случае данные iButton можно включить в дополнительный заголовок API с помощью команды ***AT$APIOPT***, в которой седьмой параметр равен 1. Мертвое время срабатывания кнопки iButton задается командой ***AT$IBTN=<sticky>,<clear>***, где ***<sticky> = 1*** означает мгновенную запись данных кнопки в память модема, а ***<clear> = 1*** означает мгновенное стирание данных кнопки из памяти модема.

Следует подчеркнуть, что данные персонального устройства iButton удаляются из памяти модема в режиме Low Power Sleep (LPS). При выполнении команды ***$OFF*** в бите Bit 12 (0x00002000) передается команда для стирания последнего считанного идентификационного номера iButton из памяти модема.

Список кнопок, разрешенных для использования конкретным автомобилем (White List), определяется командой ***AT$BTNLST***. Этот список может состоять из 150 пользователей. С устройством iButton можно работать как с использованием такого списка, так и без него. Использование списка разрешается командой ***AT$BTNCFG***.

Простейший вариант скрипта для работы с кнопкой iButon выглядит следующим образом:

AT$APIOPT=1,1,4,0,0,0,1 — информация iButton включена в расширенный заголовок с помощью седьмого параметра;

AT$BTNLST=1,»0000137453ea» — информация о кнопке занесена в White List;

AT$BTNCFG=1,0 — разрешено использование списка с задержкой 0 с;

AT$IBTN=1,1,5 — сохранять регистрационный номер кнопки в памяти модема и стирать его из ОЗУ втечение 10 с;

AT$EVENT=15,1,55,1,2 — входное событие, регистрирующее считывание данных iButton;

AT$EVENT=15,3,40,30,3014655 — выходное событие, отправка сообщения на сервер.

Подробно работа с устройствами iButton рассмотрена в [10].

Следует отметить еще одну особенность ПО серии MT4000, связанную с неотправленными сообщениями. В новых моделях значительно расширены функции сохранения сообщений [3]. Сообщения, генерируемые **Event Engine** и пересылаемые на сервер, сохраняются во flash-памяти и индексируются через лог-файл. Этот файл представляет собой журнал регистрации сообщений, в котором фиксируются обнаруженные ошибки или метаданные, связанные с сообщением. Количество сохраненных сообщений ограничено объемом памяти и зависит от размера самого сообщения. Например, если сохраненное сообщение содержит только данные NMEA, то лог-файл может содержать до 2500 таких сообщений. Сообщения бинарного формата значительно меньше по объему. Поэтому в одном файле может быть сохранено до 3000 сообщений бинарного формата. В тех случаях, когда память переполняется, новый лог-файл записывается поверх старого. Функция сохранения лог-файлов активизируется с помощью шестого бита маски событий. Если этот бит равен единице, то сообщение сохраняется в памяти модема, вне зависимости от текущего состояния регистрации модема в сети. Если шестой бит равен нулю, то предварительно тестируется статус регистрации в сети. Если регистрация в сети успешная, то сообщение пересылается сразу. Если связь отсутствует, то сообщение сохраняется в памяти модема и будет отправлено на сервер после того, как будет восстановлена связь. В памяти сохраняются только те сообщения, которые имеют шестой бит в bit mask. Служебные сообщения типа wakeup messages не имеют этой строки и поэтому не могут быть сохранены в памяти.

С помощью команды ***AT$MSGLOGEN*** пользователь может настроить модем для хранения в своей энергонезависимой памяти данных о генерируемых событиях.

Эти данные в дальнейшем можно передавать на удаленный сервер по Интернету:

```
AT$MSGLOGEN=<setting>
```

Возможные значения параметра — «0» или «1». Если выбрано значение «0», то информация о событиях не сохраняется и пересылается на сервер сразу после их поступления. Если выбрано значение «1», то данные о событиях сохраняются и могут быть прочитаны с помощью команды ***AT$MSGLOGRD***, позволяющей работать с четырьмя лог-файлами. Структура этой команды ***AT$MSGLOGRD=<queue>,<number of messages>,<starting index>,*** где:

- ***<number of messages>*** — общее количество сообщений, которые нужно читать из памяти (максимум 65535);

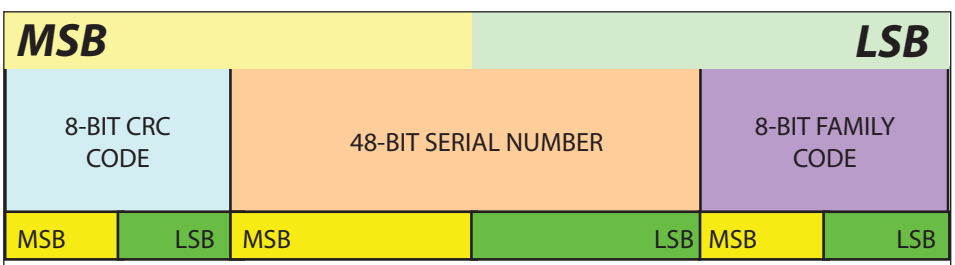

Рис. 9. Структура данных iButton в UDP-сообщении для случая, разрешенного битовой маской **<Param2>** = 0x80080000)

- *<starting index>* — стартовый индекс сообщения, сохраненного в памяти;
- *<queue>* — способ передачи SMS, UDP, TCP, USSD или различные комбинации этих протоколов (всего 43 варианта).

Например, если в памяти есть 50 непрочитанных сообщений, а общее количество сообщений равно 100, то это означает, что первые 50 сообщений были прочитаны, а последние 50 — нет. Если пользователь отправляет команду ***AT$MSGLOGRD = 0,1,51,*** будет прочитано из памяти одно сообщение с индексом 51 и затем передано на удаленный сервер. В результате общее количество непрочитанных сообщений уменьшится до 49.

С помощью команды ***AT$MSGSND*** можно выбирать способ передачи сообщения. Параметры этой команды означают следующее: *<destination>* — способ передачи (UART, SMS, UDP, TCP, PAD); *<data>* — любые 50 байтов информации в формате ASCII. Например, команда ***AT$MSGSND=0, «Hello»*** выведет сообщение «Hello» на последовательный порт.

Для отладки модема используется команда ***AT$MSGLOGDMP***, которая позволяет выводить отправленные и неотправленные сообщения на последовательный порт. Структура команды — ***AT$MSGLOGDMP=<queue>,<format>, <bytes_per_line>,<display_all>***, где:

- *<queue>* — способ передачи (UART — «0», SMS — «2», TCP — «3»);
- *<format>* — формат сообщения (ASCII — «0», HEX — «1»);
- *<bytes_per_line>* — количество байтов в сообщении;
- *<display_all>* — «0» — только неотправленные сообщения, «1» — все сообщения.

Эта команда была разработана в первую очередь в качестве утилиты поиска и устранения неисправностей при отладке модема. Так, например, с помощью этой команды можно записывать все события, включая GPS-данные, в памяти модема, не отсылая их на сервер. Данные, собранные для различных режимов вождения, затем выгружаются в ПК и анализируются. Следует учитывать тот факт, что для того, чтобы NMEA-данные сохранялись в буферной памяти, необходимо с помощью команды ***$GPSCMD*** запретить отправку этих данных на сервер.

Количество и вид неотправленных сообщений задаются командой ***AT$MLQSIZ***.

Отменить функцию сохранения, очистить журнал и отправить все данные на сервер можно с помощью команды ***AT$MSGLOGEN=0.***

В заключение следует подчеркнуть, что, в отличие от большинства GPS/ГЛОНАСС-трекеров, предлагаемых на российском рынке, модемы Novatel от Enfora могут пересылать на центральный сервер, кроме простых NMEA-сообщений, еще 214 видов UDP- и TCP-сообщений с различными параметрами движения автомобиля. Пользователь самостоятельно может выбрать необходимый ему набор параметров и типов сообщений в зависимости от стоящей перед ним задачи.

Такой подход предоставляет разработчикам широкие возможности для построения самых различных устройств. Модемы серии MT4000 можно использовать не только для спутникового мониторинга транспорта, но также и в страховом бизнесе, в системах контроля любого перемещения банкоматов, в системах контроля состояния контейнеров в морских перевозках и др.

Литература

1. Алексеев В. Новый 3G/GPS-модем Enfora Spider MT-4100 для систем страховой телематики // Беспроводные технологии. 2014. № 2.
2. UMT2202UG001 – MT 4000. User's Guide. 2012.
3. UMT2202AT001, MT4100, AT Command Reference. V. 1.02. February, 2014.
4. GSM0308UG001, Enfora GSM/GPRS Family API Reference.
5. https://support.gurtam.com
6. http://filezilla-project.org/
7. http://filezilla.ru/documentation/Using
8. Managing Script Files in Enfora Devices. Ap. Note ENF0000AN005. V. 1.00. October, 2011.
9. User Variables Overview Technical Note ENF0000TN001. Rev. 1.0. 2009.
10. ENF0000AN018, 1-Wire Interface. V. 1.02. October, 2013.