ZigBee-модули XBee:
новые возможности

Компания Digi выпустила новые версии популярных модулей XBee с предустановленным программным обеспечением, отвечающим требованиям новейшего стандарта ZigBee Pro. Применение модулей XBee упрощает и существенно сокращает время разработки систем передачи данных. В статье рассмотрены новые возможности и практические аспекты работы с модулями XBee в режиме API. Приведена схема построения беспроводного датчика температуры температуры и описан алгоритм получения информации с удаленного узла.

Олег Пушкарев

ZigBee-модули XBee являются готовыми к применению устройствами, способными самостоятельно объединяться в беспроводные сети с различной топологией — точка-точка, звезда, MESH (ячеистая сеть). Подключенный к сети модуль обеспечивает передачу данных на любой другой узел сети или на все узлы одновременно. Передача данных осуществляется по интерфейсу UART от внешнего хост-процессора, в качестве которого может выступать персональный компьютер или простейший 8-разрядный микроконтроллер стоимостью менее $1. Благодаря встроенному ПО все операции по формированию сети, присоединению новых устройств, прокладке оптимальных маршрутов сообщений осуществляются автоматически, без участия внешнего микроконтроллера. В настоящий момент компания Digi предлагает для модулей XBee несколько вариантов сетевых протоколов — это стандартные стеки ZigBee-2006, ZigBee Pro и фирменный протокол DigiMesh.

Рис. 1. Модули XBee

Сетевой протокол DigiMesh

Сетевой протокол DigiMesh был выпущен компанией Digi осенью 2008 года в виде прошивок (firmware) для модулей XBee 802.15.4, построенных на базе радиочастотной микросхемы Freescale MC13213. Уникальной особенностью протокола DigiMesh является возможность построения сети со спящими ретрансляторами (роутерами). В сети DigiMesh нет координатора с выделенной ролью — каждый из узлов сети может взять его функции на себя. Прокладка и восстановление маршрутов в данной сети осуществляется автоматически. Подобное построение гарантирует прохождение информации при выходе из строя любого узла, так как в сети DigiMesh нет «слабого звена», отказ которого мог бы привести к полной неработоспособности системы. Структурная схема сети DigiMesh приведена на рис. 2.
Преимущества сети DigiMesh по сравнению с ZigBee заключаются в наличии спящих роутеров (в ZigBee спящий режим возможен только для конечных устройств), более простом процессе запуска и большей гибкости при расширении сети. Кроме того, реализация протокола DigiMesh требует меньших ресурсов памяти микроконтроллера. Возможность перевода роутеров в режим сна обусловлена реализованным механизмом временной синхронизации всех узлов сети. В качестве маяка выступает координатор — один из узлов сети, назначенный разработчиком. Если он выходит из строя, его функции начинает выполнять любой другой узел сети. Протоколом предусмотрены

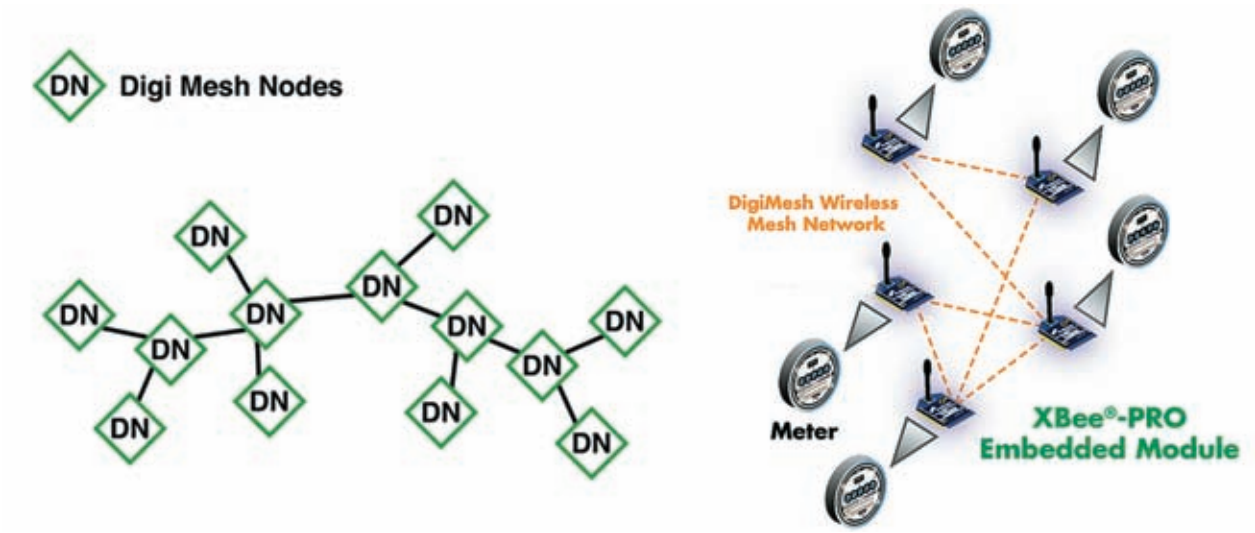

Рис. 2. Структурная схема сети DigiMesh

специальные механизмы «Номинирования» (Nomination) и «Выборов» (Election), которые позволяют разрешать коллизии, если сразу несколько узлов сети пытаются взять на себя функции координатора. С помощью команд конфигурирования можно определить узлы, которые будут иметь преимущества при «выборах» координатора. Подключение новых узлов к сети выполняется следующим образом — необходимо включить новый узел и поместить его в зоне действия сети. После первого включения новый узел будет постоянно включен на прием, ожидая синхронизирующего пакета координатора. В полученном широковещательном пакете синхронизации содержится информация о временных параметрах сети — временах сна и бодрствования, что позволяет новому узлу перейти в режим сна до следующего сеанса синхронного обмена информацией с другими узлами сети.

Стек протоколов ZigBee-Pro

Компания Digi выпустила новое программное обеспечение на базе стека протоколов EmberZNet PRO 3.1. Новое программное обеспечение позволяет строить беспроводные сети, отвечающие требованиям спецификации ZigBee-2007 (ZigBee-Pro). Новое ПО (ZB firmware) можно скачать с сайта Digi и загрузить с помощью бесплатной программы X-CTU в модули XBee ZNet 2.5 (XBee Series 2). В зависимости от функционального назначения узла и типа управления необходимо выбирать соответствующую версию прошивки (таблица).

Таблица. Функционал программного обеспечения

Версия FW	Функционал
2020	AT Coordinator
2120	API Coordinator
2220	AT Router
2320	API Router
2820	AT End Device
2920	API End Device

Особенности ПО ZB версии 2x20:

- поддержка режимов API и AT;
- спящие конечные устройства с пробуждением по линии вывода или таймеру;
- увеличенное число ретрансляций для адресных передач (команда NH);
- расширенный идентификатор 64 бит PAN ID (команды EI, OE);
- конфигурирование профиля ZigBee (команда ZS);
- удаленное конфигурирование модулей в режиме API;
- доступ к уровню ZigBee APS layer с помощью API-фрейма 0x11. Установка значений полей «APS endpoint», «cluster ID», «profile ID»;
- поддержка ZigBee-безопасности. Задание ключей шифрования «Network» и «link».

Тем, кто разрабатывает приложения на базе стека протоколов ZNet 2.5, нет необходимости в обязательном порядке обновлять программное обеспечение. Компания Digi продолжает выпуск модулей XBee ZNet 2.5, основанных на ПО EmberZNet 2.5. Следует учитывать, что модули XBee с новой прошивкой (ZB) будут не совместимыми с модулями XBee ZNet 2.5. Новая прошивка позволяет создавать ZigBee-устройства, которые могут взаимодействовать с ZigBee-узлами любых других производителей, если последние отвечают версии спецификации ZigBee-Pro.

Управление модулем в режиме API

Для управления модулем XBee и передачи данных предусмотрено два режима взаимодействия: с помощью AT-команд (прозрачный режим) или с помощью структурированных пакетов (API-режим). В некоторых случаях XBee-модули могут работать как абсолютно самостоятельные узлы, без применения внешнего управляющего устройства. Например, модуль может самостоятельно, в автоматическом режиме передавать по радиоканалу состояние входных цифровых портов и отсчеты встроенного АЦП. Управлять модулем, который работает без внешнего микроконтроллера, можно с помощью команд, передаваемых по эфиру любым другим узлом сети. Режим AT-команд очень прост для понимания и оптимален при передаче непрерывных потоков данных в сетях с простой топологией. Недостатком AT-команд является медленный переход из прозрачного режима передачи данных в режим приема модулем команд управления. Режим API гораздо более гибок, особенно при управлении с помощью внешнего микроконтроллера, так как позволяет передавать и данные, и команды управления в общем потоке. Кроме того, некоторые возможности в режиме AT-команд просто недоступны. Например, послать по ZigBee-сети AT-команду удаленному модулю можно только в API-режиме. Работа с API-пакетами требует вычисления контрольных сумм, что не очень удобно при ручном формировании пакета в окне программы X-CTU (рис. 3), однако не представляет большой сложности при управлении XBee-модулем с помощью внешнего микроконтроллера.

В качестве практического примера рассмотрим подачу обычной AT-команды ND в API-режиме. Эта команда позволяет обнаружить все узлы в сети. Команду ND можно подавать не только с координатора, но и с любого другого узла сети. При подаче этой команды узел отсылает широковещательное сообщение, которое транслируется другими узлами сети. В теории, данное широковещательное сообщение должно достигнуть всех узлов сети. При получении сообщения узел отправляет информацию о себе — свой сетевой и 64-битный адрес, тип устройства (координатор, роутер или конечное устройство), имя устройства в виде текстовой

строки (если задано) и некоторые другие свои параметры. Для того чтобы ответные сообщения не мешали друг другу, каждый обнаруженный узел делает случайную задержку перед отсылкой ответа. Поэтому при каждой отсылке команды ND ответы от обнаруженных узлов будут поступать в разном порядке. Максимальное время задержки на ответ узла по умолчанию составляет 6 с. Ниже приведены API-фреймы, полученные на выходе UART модуля XBee ZNet 2.5, с которого был отправлен запрос на обнаружение всех узлов в сети. Всего было получено четыре ответа. Как видно из приведенных ответов, в сети было обнаружено два роутера и два конечных узла.

Разберем подробнее отсылаемую AT-команду ND и полученные ответы от узлов сети:

1. Отсылаемый API-фрейм «AT-команда ND» (данная последовательность байт подается через UART на XBee-модуль):

```
7E 00 04 08 11 4E 44 54
```

Расшифровка структуры подаваемого API-фрейма:

7E	— стартовый разделитель (признак начала API-фрейма)
00 04	— длина API-фрейма
08	— ID-номер API-фрейма (подача AT-команды)
11	— номер фрейма (выбран произвольно)
4E 44	— ASCII – имя AT-команды, на которую пришел ответ (здесь: «ND»)

2. Полученные ответы (данная последовательность байт выдается на выход UART XBee-модуля)

```
7E 00 19 88 11 4E 44 00 E1 90 00 13 A2 00 40 0A 0C 22 38 00
FF FE 01 00 C1 05 10 1E 0C
7E 00 1D 88 11 4E 44 00 08 55 00 13 A2 00 40 0A 0D 4F 4E 4F
44 45 32 00 FF FE 01 00 C1 05 10 1E D2
7E 00 19 88 11 4E 44 00 9D FE 00 13 A2 00 40 0A 0B 4C 33 00
FF FE 01 00 C1 05 10 1E BE
7E 00 22 88 11 4E 44 00 25 73 00 13 A2 00 40 0A 0F 47 20 53
4C 45 45 50 5F 53 4D 41 00 25 71 02 00 C1 05 10 1E 82
```

*Расшифровка структуры фрейма ответа (для первой строки)

7E	— стартовый разделитель (признак начала API-фрейма)
00 19	— длина API-фрейма
88	— ID-номер API-пакета (ответ на AT-команду)
11	— номер фрейма, на который получен ответ
4E 44	— ASCII – имя AT-команды, на которую пришел ответ (здесь: «ND»)
00	— байт статуса (00 — OK)
E1 90	— 16-битный сетевой адрес MY
00 13 A2 00	— 64-битный уникальный адрес SH
40 0A 0C 22	— 64-битный уникальный адрес SL
38 00	— ASCII-строковый идентификатор узла NI (здесь: «3», строка заканчивается 00)
FF FE	— адрес «родительского» узла. Действительно только для конечных устройств
01	— тип устройства (0 — координаор, 1 — роутер, 2 — конечное устройство)
00	— статус
C1 05	— профиль устройства (profile ID)
10 1E	— производитель (manufacturer ID)
0C	— контрольная сумма

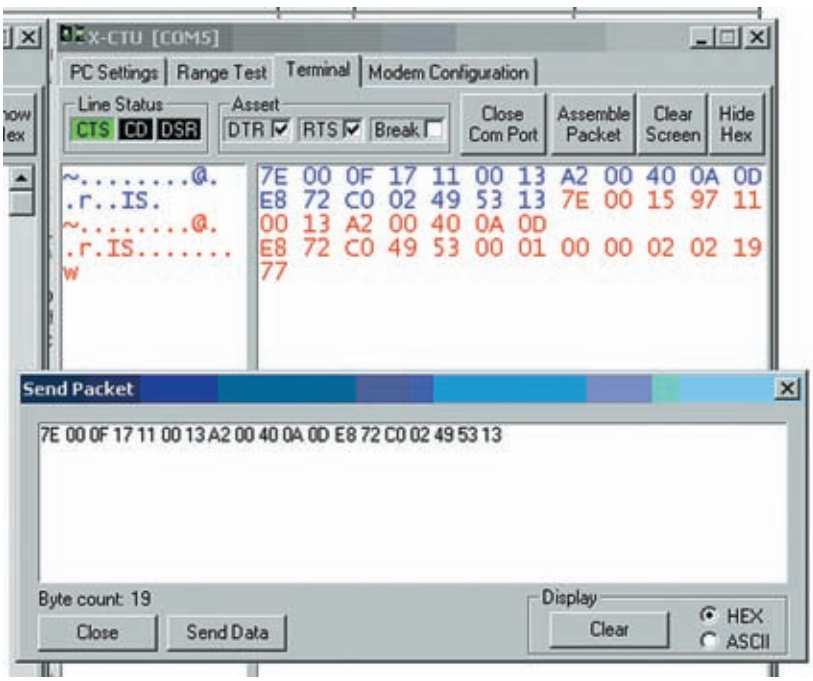

Рис. 3. Примеры работы в API-режиме в программе X-CTU

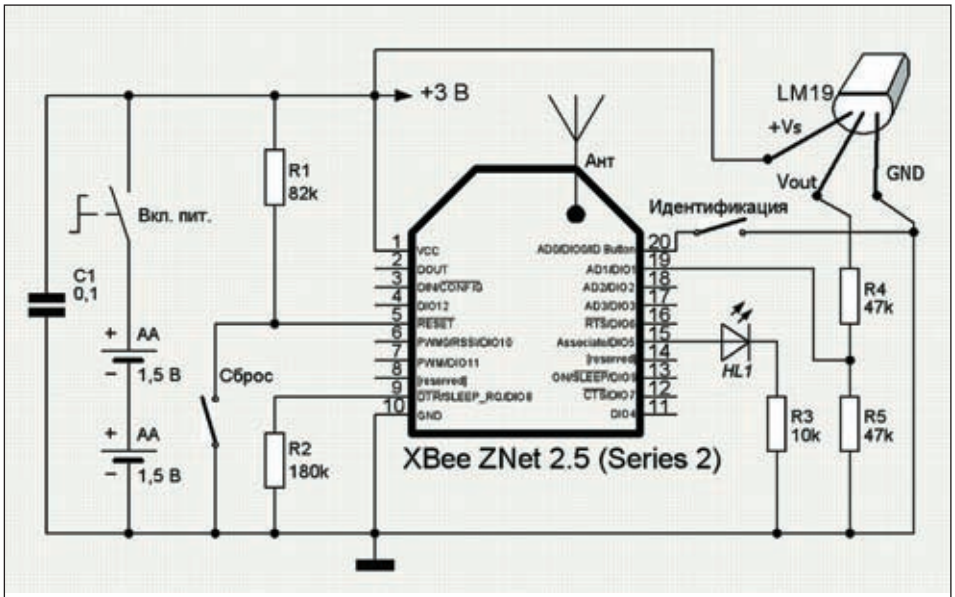

Рис. 4. Принципиальная схема температурного датчика

Беспроводной датчик температуры

В качестве примера работы модуля XBee без использования встроенного микроконтроллера рассмотрим вариант построения беспроводного датчика температуры. Для удаленного измерения температуры воспользуемся возможностью удаленной отсылки API-команды по эфиру. Схема измерительного модуля довольно проста и приведена на рис. 4. В качестве измерителя температуры использован аналоговый интегральный датчик LM19. Микросхема LM19 преобразует температуру в диапазоне от –55 до +130 °C в выходное напряжение, которое можно измерить с помощью встроенного в модуль XBee АЦП. В связи с тем, что диапазон выходного напряжения LM19 (0,303–2,485 В) превышает максимальное измеряемое напряжение АЦП модуля (1,2 В), в схеме применен делитель напряжения на резисторах, понижающий выходное напряжение LM19 в 2 раза. Рассматриваемый беспроводной датчик (рис. 5) температуры работает от двух батарей типа «АА». Светодиод HL1 индицирует режим работы модуля: светится постоянно — модуль не присоединился к ZigBee-сети, мигает два раза в секунду — произошло присоединение модуля к ZigBee-сети.

Для проверки работоспособности беспроводного датчика температуры развернем простейшую ZigBee-сеть из двух узлов — координатора и беспроводного датчика. В качестве коорди-

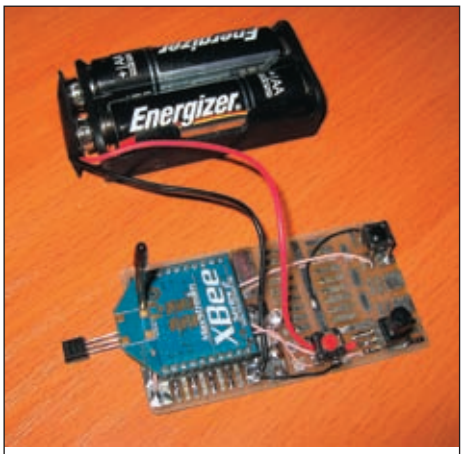

Рис. 5. Температурный датчик на базе модуля XBee

натора будем использовать XBee-модуль, установленный на переходную плату из комплекта отладочного набора XB24-BPDK и подключенный к ПК с помощью кабеля USB или RS-232. До начала эксперимента необходимо сделать следующие настройки для XBee-модулей с помощью закладки установок параметров в программе X-CTU:

1. XBee-модуль координатора должен иметь прошивку v. 1147 ZNET 2.5 COORDINATOR API.
2. XBee-модуль датчика температуры должен иметь прошивку v. 1347 ZNET 2.5 ROUTER/END DEVICE API.
3. Задать текстовое имя для XBee-модуля датчика температуры, например 12345 (команда NI).
4. Разрешить работу АЦП для XBee-модуля датчика температуры на выводе AD1 (команда D1=2).

В тестовом проекте будем с помощью координатора, подключенного к ПК, отправлять запросы и получать пакеты, содержащие информацию о температуре удаленного узла. Для простейшего эксперимента сначала ограничимся простейшей сетью из 2 узлов — только координатор и температурный датчик. Датчик температуры будет работать постоянно, то есть без использования спящего режима. После включения XBee-модулей произойдет автоматическое подключение температурного датчика к координатору. При этом на выходе UART координатора через закладку Terminal программы XCTU можно наблюдать API-фрейм идентификации с информацией о параметрах подсоединенного узла (рис. 7). Данный фрейм мы также будем получать на выходе координатора при нажатии кнопки «Идентификация» на температурном датчике.

Для отправки запроса на удаленное считывание значения АЦП (температуры) необходимо знать 64-разрядный уникальный и 16-разрядный сетевой адрес удаленного узла. Эти значения можно взять из полученного сообщения о присоединении (00 13 A2 00 40 0A 0F 47 и 22 62 соответственно). Теперь можно сформировать API-фрейм на удаленное считывание АЦП. Контрольную сумму для отправляемых пакетов в данном случае рассчитываем вручную. После отправки пакета мы получим ответный API-фрейм (рис. 6), содержащий значение напряжения АЦП, которое можно пересчитать в температуру. Для преобразования аналого-цифровых данных в милливольты используйте следующий алгоритм:

$$AD\ (мВ) = \\ = (значение\ ADIO/0x3FF) \times 1200\ мВ$$

Из полученного пакета видим, что значение АЦП = 02 8F (предпоследние 2 байта в API-фрейме), что в абсолютном исчислении составляет 0,768 В. С учетом делителя напряжение на выходе LM19 было 0,891 × 2 = 1,536 В, что соответствует температуре 24,5 °С. Формулы для пересчета выходного напряжения в значения температуры можно найти в документации на LM19.

По мере освоения API-команд можно добавлять в сеть новые узлы и тестировать передачу данных с ретрансляцией пакетов. Для минимизации энергопотребления модуль можно перевести в спящий режим (SM = 4) с периодическим пробуждением по встроенному таймеру, например каждые 3 с (SP = 0x12C, ST = 0x1F4). Тогда модуль будет периодически просыпаться, отсылая запрос «родительскому» узлу на проверку наличия информации для себя. В этом случае рекомендуется задать с некоторым запасом время хранения сообщений на родительском узле, например SP = 0x2BC (7 с).

Заключение

Предлагаемые компанией Digi различные сетевые протоколы позволяют строить беспроводные сети, оптимизированные под конкретные требования пользовательского приложения. Благодаря упрощенному интерфейсу управления, практическое построение беспроводной сети на базе модулей XBee не требует глубоких знаний ZigBee-спецификации и может быть выполнено в сжатые сроки разработчиком средней квалификации.

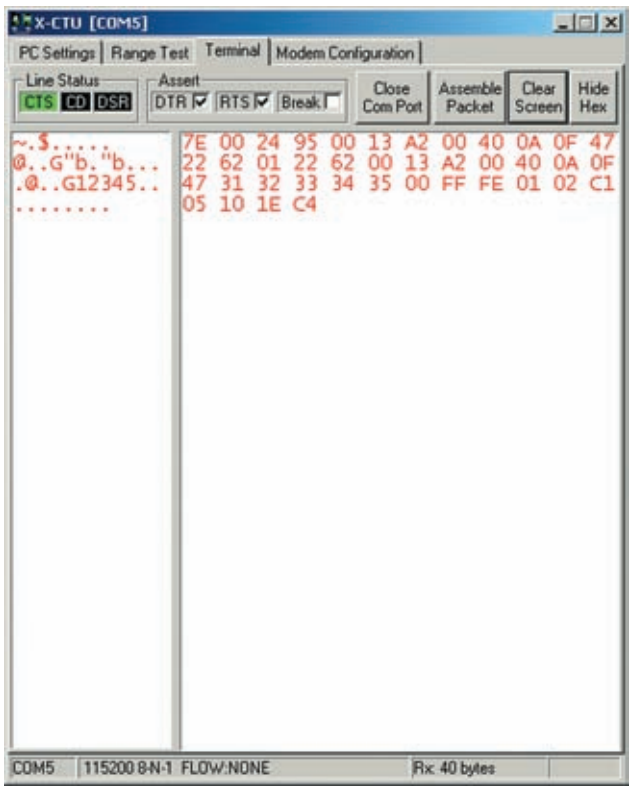

Рис. 6. API-фрейм с параметрами присоединенного узла сети

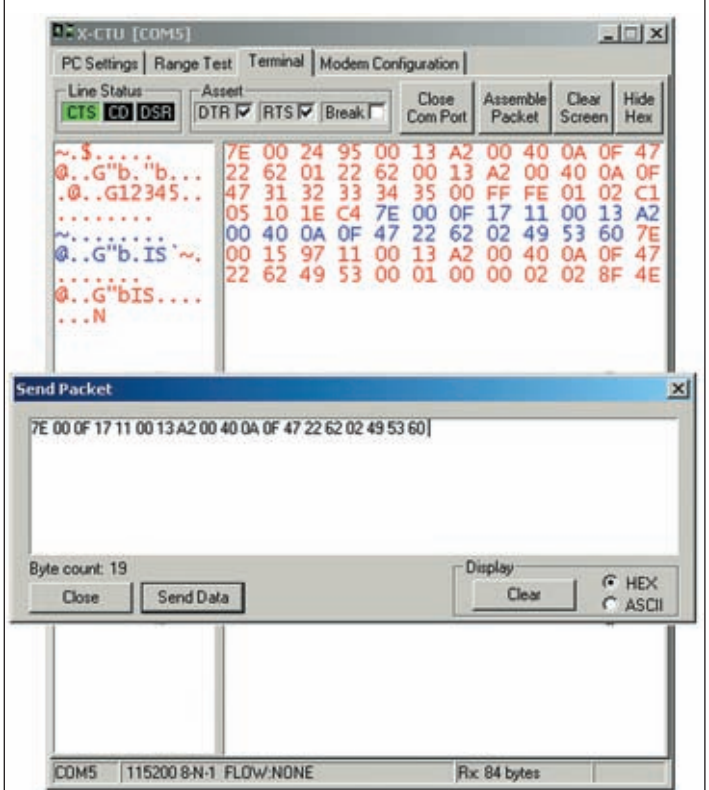

Рис. 7. Закладка Terminal программы XCTU