

Программный драйвер для управления GSM-модулем SIM800C

В статье представлен краткий обзор интерфейса управления бюджетным GSM-модулем SIM800C через последовательный порт с помощью AT-команд, описаны основные плюсы и минусы такого интерфейса. Предоставлен кросс-платформенный драйвер с открытым исходным кодом для преобразования интерфейса AT-команд в интерфейс, состоящий из простых функций управления модулем, контроля его состояния, отправки и получения данных.

Данная информация может быть полезной начинающим разработчикам систем управления и контроля с использованием технологии GSM.

Сергей Гаевский
Email: info@nuvoton.com.ua

Сегодня промышленная и бытовая автоматизация немыслимы без использования средств беспроводной передачи данных, позволяющих объединять отдельные устройства в сети, предоставляющие как локальный, так и глобальный доступ к элементам управления и контроля отдельных модулей и системы в целом. Среди великого множества беспроводных сервисов первое место по праву занимает сотовая связь с ее 20-летним стажем на отечественном рынке телекоммуникационных услуг. Практически полное покрытие территории, сравнительно низкие и гибкие тарифы, множество дополнительных сервисов позволяют эффективно использовать мобильные сети в большинстве промышленных и бытовых применений: системах телеметрии, удаленного управления, отслеживания перемещения подвижных объектов и т. п.

Одной из самых старых и востребованных технологий сотовой связи остается технология GSM, разрешающая выполнять голосовые звонки между устройствами, обмениваться

короткими текстовыми сообщениями и обеспечивающая мобильный доступ в глобальную сеть Интернет.

Массовый спрос на GSM-оборудование привел к его значительному удешевлению по сравнению с устройствами, используемыми 3–5 лет назад. Компании — производители чипсетов разработали дешевые интегрированные миниатюрные решения, которые значительно упрощают схемотехнику за счет уменьшения количества внешних элементов, необходимых для функционирования таких устройств. Примером может служить чипсет MT6261 от компании MediaTek, первоначально разработанный для «умных» часов и других миниатюрных гаджетов, но затем нашедший широкое применение в GSM-модулях — полуфабрикатах, предназначенных для инсталляции в бытовые и промышленные устройства, использующие различные сервисы GSM-сети: охранные сигнализации, трекеры, элементы системы «умный дом» и т. д.

Наибольшую популярность среди отечественных разработчиков получила «массовая» серия модулей SIM800C от компании SIMCom (рис. 1). При розничной цене менее \$4 данные модули обеспечивают широкий функционал, имея лучшее соотношение возможностей и стоимости в своей товарной нише. Несомненным преимуществом модулей SIM800C является наличие множества готовых решений с открытым исходным кодом (модули распространены в среде разработчиков под платформу Arduino), а также эффективная техническая поддержка поставщика, тесно сотрудничающего непосредственно с командой разработчиков компании SIMCom.

Модуль SIM800C представляет собой изделие-полуфабрикат размером около 16×18 мм, которое предназначено для поверхностного монтажа

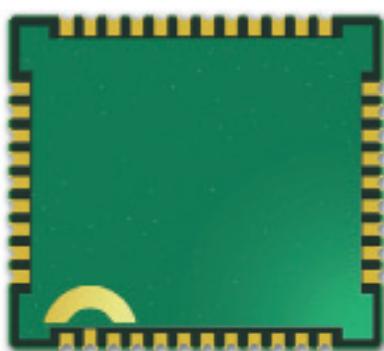


Рис. 1. GSM-модуль SIM800C

на печатную плату. По периметру расположено 42 вывода с шагом 1,1 мм, что позволяет запаять модуль даже вручную. Модуль содержит интерфейсы подключения SIM-карты, аналоговых звуковых цепей, USB, UART и цифровые входы/выходы общего назначения (рис. 2). Питание осуществляется от стабильного источника напряжением 3,5–4,2 В, в качестве которого может выступать стандартный литиевый аккумулятор. Следует отметить, что логические уровни единицы на интерфейсных выводах модуля составляют 2,8 В, поэтому пользователь должен позаботиться о согласовании с логическими уровнями микроконтроллера, которые обычно равны 3,3 или 5 В. Потребляемый от источника питания ток в среднем составляет 100–200 мА (в режиме ожидания — несколько миллиампер), но импульсные токи могут достигать 2 А, поэтому для предупреждения кратковременной просадки напряжения питания необходимо применять низкоомпедансные блокировочные конденсаторы, размещая их в непосредственной близости от модуля. В случае использования блока питания следует обеспечить запас по мощности во избежание сбоев во время работы, особенно в местах со слабым сигналом сотовой сети. Более подробно особенности проектирования устройств с применением GSM-модуля SIM800C описаны в документе [1].

Антенные цепи модуля имеют импеданс 50 Ом в рабочих диапазонах 900 и 1800 МГц. К сожалению, GSM-антенны, доступные в масовой продаже, а также суррогатные антенны, выполненные как в виде проводников, так и печатным монтажом, могут иметь весьма существенные отклонения от заданного импеданса, а иногда и значительную реактивную составляющую на одном из GSM-диапазонов. В таких случаях усилитель мощности модуля, обеспечивающий до 2 Вт полезного сигнала, будет работать с перегрузками, что может привести к выходу его из строя. Также неизбежно повысится уровень гармоник в выходном сигнале, что становится препятствием для сертификации конечного устройства. Для предотвращения нежелательных последствий рекомендуется предусмотреть элементарные цепи согласования между модулем и антенной в соответствии с рекомендациями производителя, приведенными в технической документации.

Для функционирования модуля в составе пользовательского устройства необходим интерфейс, позволяющий управлять модулем и получать информацию о его текущем состоянии, а также отправлять данные удаленному объекту и принимать их. Производители использовали два подхода к управлению модулем: с помощью API-функций, реализуемых пользователем во внутренней прошивке модуля, и с помощью AT-команд, посыпаемых с внешнего устройства (в том числе с терминалом в ручном режиме), подключенного к модулю через стандартный серийный интерфейс (UART, COM-порт).

Управление с использованием API (так называемая технология Embedded AT [2]), несомненно, намного эффективнее и позволяет создать более простой код, а также избавиться от трудностей, связанных с работой UART и разбором текстовых строк. Недостатком данной технологии является необходимость двух

наборов утилит: для разработки, компиляции и программирования пользовательского кода, помещаемого во флэш-память самого модуля, и для кода, выполняемого во внешнем микроконтроллере. Полностью отказаться от внешнего микроконтроллера удается лишь в редких случаях, поскольку операционная система внутри модуля не является системой реального времени, что накладывает ограничения на создание пользовательских физических интерфейсов, отсутствующих в виде аппаратной реализации в модуле и чипсете. Идеальным видится применение бюджетного внешнего контроллера реального времени для обеспечения физических интерфейсов, в то время как основная логика функционирования устройства реализуется в коде, выполняемом в самом модуле.

Управление с помощью AT-команд является более простым и общепринятым решением, которое рекомендуется начинающим разработчикам. Этот интерфейс хорошо документирован [3] и обеспечивает доступ к плавающему большинству функций модуля, включая некоторые аудиовозможности и доступ к внутренней файловой системе. Его основной недостаток — отсутствие стандартизации AT-команд даже в пределах одной модели модуля. Так, в различных версиях прошивки могут быть незначительные отличия в реализации той или иной команды, несущественные при правильном программировании, но дающие нежелательные эффекты в сочетании с нестандартными алгоритмами работы пользователя.

Кроме того, следует отметить, что набор AT-команд первоначально разрабатывался для интерфейса «машина-человек» и идеален для управления модулем через терминал в ручном режиме. При попытке реализации на базе AT-команд интерфейса «машина-машина» (управление модулем с помощью внешнего микроконтроллера) разработчик неизбежно сталкивается с трудностями, связанными с созданием подходящей под его задачу машины состояний и интерпретацией текстовых строк, возвращаемых модулем. Это значительно усложняет разработку программного обеспечения и требует дополнительных вычислительных ресурсов. Кроме того, количество ошибок интерфейса обычно значительно превышает число ошибок в логике самой программы (особенно в случае несложных бюджетных устройств), процесс отладки требует значительных ресурсов и соответствующего уровня программиста, что заметно повышает стоимость такого программного обеспечения, и, как результат, конечного устройства.

Один из вариантов решения данной проблемы — использование универсального драйвера, преобразующего интерфейс AT-команд в другой интерфейс, более подходящий для программиста, разрабатывающего код для внешнего управляющего микроконтроллера на языке Си. Конечно, создание универсального драйвера, позволяющего в полной мере реализовать все доступные через AT-команды функции модуля, становится весьма сложной

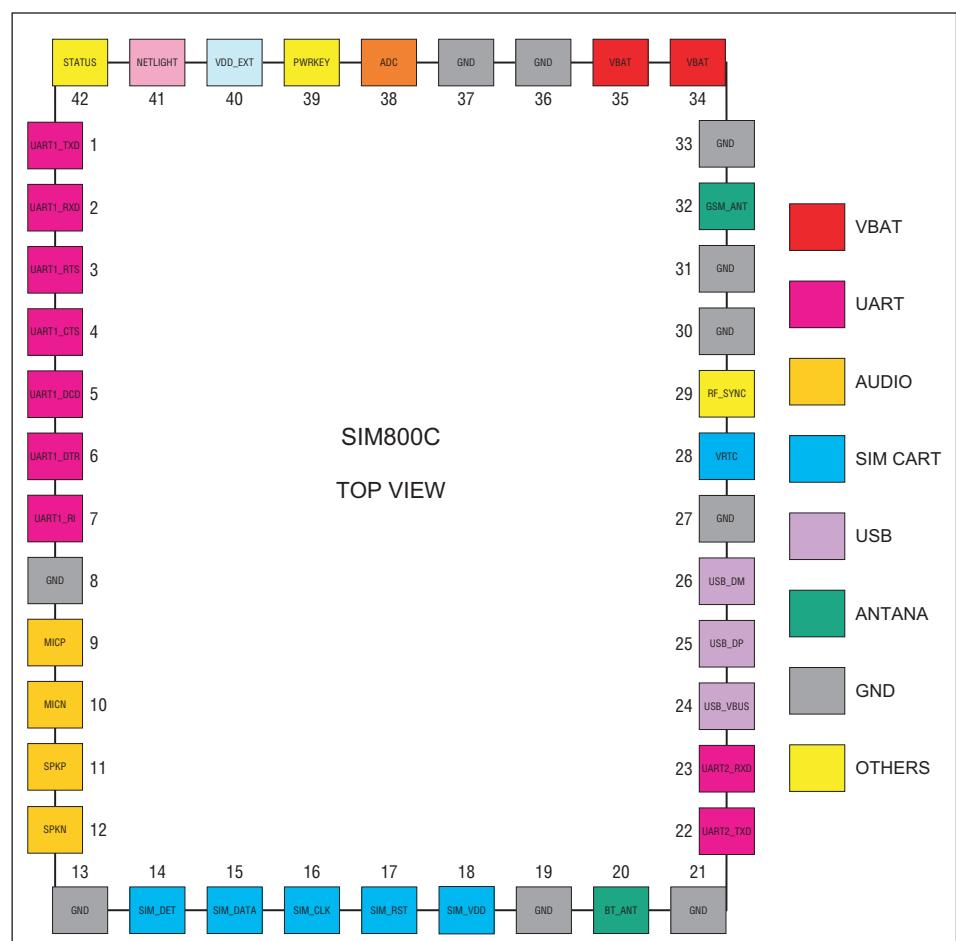


Рис. 2. Расположение выводов модуля SIM800C

Таблица 1. Интерфейсы драйвера

Функция	Описание	Связанный флаг
А. Интерфейс «драйвер-модуль»		
void rx(int8_t* data, int16_t len)	Передает драйверу 0<len<255 байт данных в data, полученных от модуля через UART. Может вызываться непосредственно с обработчика прерывания.	Флаг прерывания UART RX
int16_t tx(int8_t* data)	Читает данные в data из драйвера для отправки в UART, возвращает длину данных в байтах	STATE_TX
uint32_t poll(uint32_t time)	Выполняет обработку внутри драйвера. В качестве параметра принимает счетчик секунд. Возвращает значения флагов состояния драйвера. Вызывается периодически из основного цикла программы	
Б. Функции чтения		
int16_t lasterr(void);	Возвращает код ошибки драйвера	STATE_ERROR
void callnum(uint8_t* num)	Читает номер звонящего абонента	STATE_CNUM
int16_t readdtmf(uint8_t* data)	Возвращает количество принятых DTMF-символов и до 20 символов '0'-'F'	STATE_DTMF
int16_t readsms(uint8_t* data, uint8_t* phone)	Возвращает длину текста принятого SMS, сам текст и телефон отправителя в виде 0-терминированной строки	STATE_SMS
int16_t smsd(void)	Возвращает код состояния (подтверждение о доставке) отправленного SMS	STATE_SMSD
int16_t readdata(uint8_t* data, uint8_t* ch)	Возвращает длину двоичных данных в data, принятых по каналу GPRS, и номер сокета (0-4), принявшего данные	STATE_DATA
В. Функции управления звонками		
void call(int8_t* num)	Выполняет голосовой звонок по номеру num, заданному в виде строки в международном формате. При num==NULL завершает активный звонок	STATE_CALL
void answer(void)	Отвечает на входящий голосовой звонок	STATE_CALL
void reject(void)	Сброс входящего звонка	STATE_CALL
Г. Функции управления GPRS и сокетами		
void gprs(int8_t* apn)	Активирует GPRS с точкой доступа apn. При apn==NULL деактивирует GPRS	STATE_GPRS
void listen(uint16_t port)	Активирует TCP-сервер на порте port. При port==0 деактивирует TCP-сервер	STATE_SRV
void connect(uint8_t ch, uint8_t tcp, int8_t* dest, uint16_t port)	Подключает сокет 0<=ch<=4 к удаленному хосту с IP-адресом dest и портом port. При tcp==1 используется транспорт TCP, иначе UDP	STATE_CH0 – STATE_CH4
void close(uint8_t ch)	Отключает сокет 0<=ch<=4	STATE_CH0 – STATE_CH4
Д. Функции отправки данных		
void senddtmf(uint8_t* data, int16_t len)	Отправляет строку DTMF-символов '0'-'F' в data длиной 0<len<20. Выполняется во время активного голосового звонка	STATE_CALL
void sendsms(uint8_t* phone, int8_t* data)	Отправляет текстовое сообщение в виде стоки в data адресату phone (в международном формате). Флаг STATE_SMSD устанавливается при доставке	STATE_SMSD
void send(uint8_t ch, int8_t* data, uint16_t len)	Отправляет двоичные данные в data длиной 1<=len<=576 через сокет 0<=ch<=4. Флаг STATE_SEND устанавливается при успешной отправке	STATE_SEND
Е. Служебные функции		
void reinit(void)	Ручной запуск первичной инициализации GSM-модуля	STATE_INIT

задачей, посильной лишь производителю модулей. Кроме того, подобный универсальный драйвер потребует значительного количества ресурсов и будет весьма непрост в настройке и использовании, что в конечном итоге может свести на нет преимущества подобного решения, особенно в сравнении с технологией Embedded AT. Но реализация в виде драйвера основных функций модуля, наиболее востребованных пользователями в стандартных ситуациях, несомненно, будет полезной, особенно для начинающих разработчиков. К тому же данный подход может

значительно сэкономить время и стоимость разработки в случае проектирования массовых бюджетных решений, например систем в составе «умного дома».

Учитывая актуальность указанной проблемы, мы попытались создать собственный кросс-платформенный драйвер для модуля SIM800C, конвертирующий интерфейс AT-команд в нативный Си-интерфейс. При разработке особое внимание былоделено простоте использования и экономии ресурсов (вычислительной мощности процессора, оперативной и постоянной памяти). Таким

образом, представленный драйвер идеально подходит для недорогих микроконтроллеров с архитектурой Cortex-M0, функционирующих без операционной системы.

На самом верхнем уровне драйвер предоставляет два интерфейса: во-первых, реализован интерфейс обмена данными с модулем, во-вторых, предусмотрен набор функций, позволяющих получить уведомления о событиях модуля (смене состояния, получении и отправке данных и т. п.), а также выполнить команды управления модулем. Кроме того, обеспечивается возможность отправки и получения двоичных и текстовых

Таблица 2. Флаги состояния драйвера

Флаг	Описание	Установка	Сброс
STATE_INIT	Драйвер готов к работе	Авто	reinit()
STATE_BUSY	Драйвер занят и не может принимать функции групп BE	Функция групп BE	Авто
STATE_REG	Регистрация модуля в сети GSM	Авто	Авто
STATE_TX	Отсылка данных от драйвера GSM модулю	Авто	int16_t tx(int8_t* data)
STATE_ERROR	Ошибка драйвера	Авто	int16_t lasterr()
STATE_CALL	Активный голосовой звонок	Авто, call(int8_t* num)	Авто, call(0), reject()
STATE_CNUM	Получен номер звонящего абонента	Авто	callnum(uint8_t* num)
STATE_DTMF	Получена DTMF-посылка	Авто	int16_t readdtmf(uint8_t* data)
STATE_SMS	Получено SMS	Авто	int16_t readsms(uint8_t* data, uint8_t* phone)
STATE_SMSD	Получено подтверждение о доставке SMS	Авто	int16_t smsd()
STATE_GPRS	Состояние подключения к GPRS	gprs(int8_t* apn)	Авто, gprs(0)
STATE_SRV	Состояние TCP-сервера	listen(uint16_t port)	listen(0)
STATE_DATA	Получены данные по каналу GPRS	Авто	int16_t readdata(uint8_t* data, uint8_t* ch)
STATE_SEND	TCP-данные доставлены	Авто	send(uint8_t ch, int8_t* data, uint16_t len)
STATE_CH0 – STATE_CH4	Состояние сокетов 0-4	Авто, connect(uint8_t ch, uint8_t tcp, int8_t* dest, uint16_t port)	Авто, close(uint8_t ch)

данных по различным каналам связи (DTMF, SMS, GPRS TCP/UDP).

Интерфейс обмена данными с модулем представлен двумя функциями: функцией чтения из драйвера данных, предназначенных для отправки в модуль, и функцией отправки в драйвер данных, полученных от модуля. Последняя функция может быть безопасно вызвана из обработчика прерываний микроконтроллера и способна передавать в драйвер данные длиной 1–255 байт. Это дает возможность легко реализовать канал «модуль–драйвер» с помощью прерывания от UART RX или от DMA-микроконтроллера, передавая 1 байт или сразу блоки данных переменной длины. Обратный канал данных «драйвер–модуль» должен быть реализован в зависимости от используемой платформы, например с помощью прерывания UART TX или DMA. Драйвер может выдавать блок данных длиной до 600 байт (MTU + заголовки), поэтому пользователь должен предусмотреть буфер соответствующего размера для хранения данных в процессе их побайтной или поблочной выдачи.

Интерфейс обмена с пользовательской программой представлен двумя группами функций: чтения и записи (управления). Функции первой группы должны вызываться пользователем при наступлении определенных событий в драйвере. Они предназначены для получения данных из драйвера (например, текста SMS или данных, принятых по TCP). Функции второй группы могут быть вызваны, если драйвер свободен (не занят обработкой предыдущей команды) и используется для управления модулем и передачи данных удаленной стороне.

Отдельно выделяется функция, реализующая собственно механизм драйвера. Данная функция должна периодически вызываться из основного цикла пользовательской программы. Она возвращает флаги состояния, информирующие пользователя о событиях драйвера и определяющие его дальнейшие действия, в частности вызов нужной функции чтения или управления. Аргументом этой функции является счетчик секунд, реализованный пользователем, например, с помощью таймера. Время необходимо драйверу для отработки тайм-аутов, связанных с функционированием GSM-модуля. Описание функций драйвера представлено в таблице 1, описание флагов состояния — в таблице 2.

Таким образом, код драйвера максимально отделен от аппаратной платформы и требует от пользователя лишь реализации функции отправки блока данных в UART, обработки прерываний, возникающих при получении данных из UART (для отправки их в драйвер), и счетчика секунд на базе системного таймера. Подобный подход также предусмотрен во многих других кросс-платформенных драйверах (например, TCP-IP-стеке LWIP) и наиболее оптимален с точки зрения переносимости между платформами. Так, мы без особых проблем использовали наш драйвер на платформах Windows и ARM Linux, Cortex-M0 и M4.

Исходный код драйвера доступен под лицензией GNU LGPL на сервисе github [4], что предоставляет возможность пользователям добавлять необходимые функции, например работу с Bluetooth (профиль SPP) или применение технологии CSD в модулях, чипсеты

которых ее поддерживают. Это помогает и оперативно исправлять возможные ошибки, в том числе после очередного обновления прошивки GSM-модуля. Благодаря хорошей структурированности кода драйвер с минимальными изменениями может быть использован и с другими GSM-модулями. Так, мы бегло тестировали его с другими модулями SIMCom серии 800, функционал оказался полностью сохранен.

Для использования драйвера необходимо в проект добавить файлы if.c, ini.c, atc.c и urg.c, а в файл main.c включить заголовки if.h и def.h. Остается лишь отметить, что в основном цикле после каждого вызова функции poll необходимо анализировать флаги и выполнять только одно действие с драйвером в каждой итерации цикла.

Таким образом, данный драйвер позволяет начинающим разработчикам быстро создавать несложные устройства, использующие GSM-сервисы. Основные функции бюджетного модуля SIM800C будут доступны без глубокого изучения документации на AT-команды и написания кода разбора текстовых строк и машины состояний. ■

Литература

- [1. www.microchip.ua/simcom/SIM800x/SIM800C/SIM800C_Hardware_Design_V1.05.pdf](http://www.microchip.ua/simcom/SIM800x/SIM800C/SIM800C_Hardware_Design_V1.05.pdf)
- [2. www.microchip.ua/simcom/?link=/SIM800x/EAT](http://www.microchip.ua/simcom/?link=/SIM800x/EAT)
- [3. www.microchip.ua/simcom/SIM800x/SIM800%20Series_AT%20Command%20Manual_V1.10.pdf](http://www.microchip.ua/simcom/SIM800x/SIM800%20Series_AT%20Command%20Manual_V1.10.pdf)
- [4. www.github.com/GammaUkraine/SIM800C](http://www.github.com/GammaUkraine/SIM800C)