Embedded AT:

начало работы с технологией интеграции пользовательского ПО в GSM/GPRS-модуль SIM800/SIM800H

Батор Батуев
bator.batuev@sim.com

Введение

В статье подробно описаны шаги, которые необходимо предпринять для освоения технологии Embedded AT в GSM/GPRS-модулях SIM800H и SIM800 (рис. 1) производства SIMCom Wireless Solutions[1]. Embedded AT (далее — EAT) — проприетарная технология интеграции пользовательского программного Си-кода с ядром модуля SIM800H/SIM800. EAT позволит в некоторых случаях избавиться от внешнего микроконтроллера в конечном устройстве, что сэкономит место на печатной плате, упростит процесс ее комплектации, сборки, проверки и в итоге удешевит бюджет изделия в целом. Данная статья будет полезна схемотехникам и программистам — разработчикам M2M-устройств, таких как:

- автомобильные ГЛОНАСС/GPS-терминалы;
- автомобильные охранно-поисковые системы;
- персональные трекеры;
- охранные системы;
- счетчики газа, воды и электричества;
- концентраторы и модемы систем сбора данных и т. д.

Технические характеристики модулей SIM800 и SIM800H. Отличия от 900-й серии

Общая особенность модулей SIM800 и SIM800H — наличие встроенного блока Bluetooth и интерфейса SD-карты, а также более высокий класс GPRS (Class 12), что означает на практике более высокую скорость передачи Upload. Для сравнения в таблице 1 показаны основные технические характеристики популярного сегодня модуля SIM900R и модулей SIM800 и SIM800H. Как видно, существуют две версии SIM800 — с памятью программ 32 и 64 Мбит. 32-мегабитовая версия SIM800 программно поддерживает либо Bluetooth, либо EAT. Это относится и к SIM800H. 64-мегабитовая версия SIM800 поддерживает Bluetooth и EAT одновременно, что делает эти модули более универсальными.

а

б

Рис. 1. Модули SIMCom: а) SIM800H; б) SIM800 (вид сверху и снизу)

Важно отметить, что модуль SIM800 является частично совместимым по выводам с SIM900R. Оба модуля имеют одинаковые размеры и торцевые контакты для пайки. В таблице 2 показаны отличия в распиновке модулей SIM800 и SIM900R. Если данные отличия не коррелируют с дизайном платы на базе SIM900R, то можно сказать, что SIM800 и SIM900R в большей степени взаимозаменяемы. Единственные вопросы возникают с выводами 2 и 53 при установке SIM800 на плату для SIM900R. В SIM800 допускается не подключать землю на выводе 2, если остальные выводы земли подключены. Вывод ANT_BT допускается подключать на землю, если не применяется функция Bluetooth, даже в версиях программного обеспечения (ПО), поддерживающих Bluetooth (если блок Bluetooth отключен, ***AT+BTPOWER = 0***).

У пользователей модулей SIM900R, возможно, возникнет вопрос — совместимы ли программно модули SIM800 и SIM800H с SIM900R и не придется ли переписывать ранее отлаженный код. Модули серии 800x разработаны с учетом архитектуры и принципов работы 900-й серии. Да, в 800-й серии имеются AT-команды, свойственные только ей, но бóльшая часть команд полностью совпадает с системой команд 900-й серии. К примеру, те, кто использует в модулях SIM900R функции, TCP/IP, voice/data call и SMS, не заметят разницы в поведении модулей SIM900R, SIM800 и SIM800H. Но поскольку применение модулей SIM800 и SIM800H обуславливается, скорее, наличием в них дополнительных функций, то доработка кода все же будет неизбежна и вопрос о программной совместимости лежит в плоскости удобства и легкости освоения 800-й серии.

[1] SIMCom Wireless Solutions (www.sim.com/wm) — известный производитель беспроводных M2M GSM/GPRS-, 3G-, LTE- и GPS/ГЛОНАСС-решений для M2M-индустрии.

Таблица 1. Сравнительная таблица программно-аппаратных функций SIM800H, SIM800 и SIM900R

Наименование	SIM800H (32 Мбит)	SIM800 (32 Мбит)	SIM800 (64 Мбит)	SIM900R (64 Мбит)
Внешний вид				
Код наименования	S2-105HV-XXXXX	S2-105MB-XXXXX	S2-105MC-XXXXX	S2-1047R-XXXXX
Стандартные функции				
Voice (07.07)	+			–
SMS (07.05)	+			
GPRS Class	12 B (85,6 кбит/с Upload; 85,6 кбит/с Download)			10 B (42,8 кбит/с Upload; 85,6 кбит/с Download)
CSD	+			
MUX (07.10)	+			
Аппаратные функции				
Bluetooth 3.0 + EDR	+			–
Диапазоны GSM	Quad Band			Dual Band
SIM-карта	1			
Корпус	LGA	LCC		
Размер, мм	15,8×17,8	24×24		
Аудиоинтерфейс	2 analog + PCM	1 analog + PCM		1 analog
AT-порт + обновление ПО	UART			MAIN UART
Диагностический порт + обновление ПО	USB			DBG UART
Поддержка внешней SD-карты	+			–
Напряжение питания, В	3,4–4,4			3,2–4,8
Потребление в режиме сна, мА	1,02 (сохраняя регистрацию в сети)	1,3		1
Диапазон рабочих температур, °C	–40...+85			
Программные функции				
PPP	+			
TCP/IP and UDP/IP	+			
HTTP and FTP	+			
EMAIL (SMTP, POP3)	+			
MMS	+			
DTMF Decoding	+			
Jamming Detection	+			
GSM Location	+			
PING	+			
AMR play	+			
Сканирование сети (без SIM-карты)	+			
Bluetooth 3.0	+ (СПО*, исключает Embedded AT и CMUX)		+ (СПО, исключает CMUX)	–
Embedded AT	+ (СПО, исключает Bluetooth)			+ (СПО)

Примечание: *СПО (здесь и далее) — специализированное программное обеспечение.

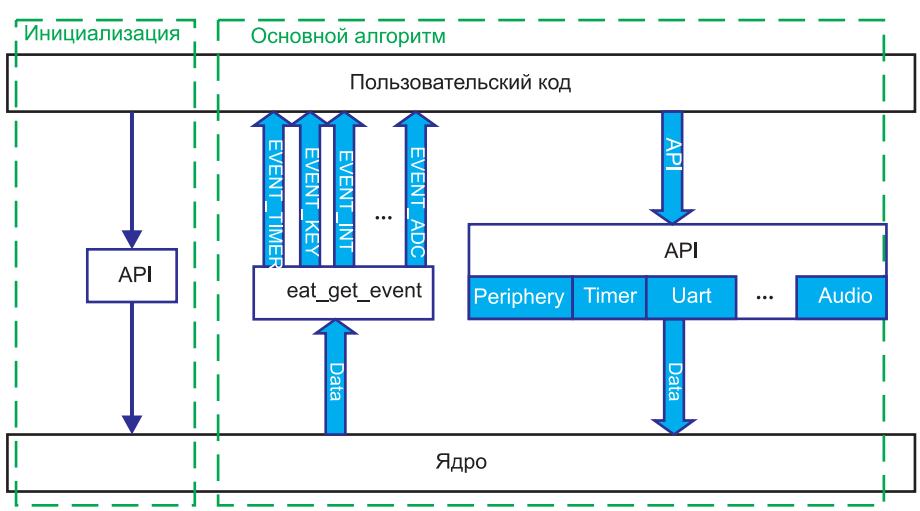


Рис. 2. Структура взаимодействия пользовательского кода и ядра ПО модуля

Общие сведения об Embedded AT

Как было сказано, EAT позволяет интегрировать Си-код пользователя в программный код модуля. Эта возможность обеспечивается архитектурой специальной версии ПО модуля (стандартное ПО не поддерживает EAT). Принцип взаимодействия пользовательского кода и ядра ПО модуля схематически изображен на рис. 2.

Для взаимодействия пользовательского Си-кода с ядром ПО модуля от SIMCom Wireless Solutions предоставляются API (Application Programming Interface, «программный интерфейс приложения»), которые дают доступ к ресурсам модуля, таким как:

- 1,5 Мбайт памяти программы и данных;
- 500 кбайт оперативной памяти;
- 29 GPIO, 5 GPIO с функцией прерывания;
- 16 таймеров;
- 1 SPI;
- 1 PWM;
- 2 UART;
- 1 USB;
- 1 ADC;
- интерфейс клавиатуры 5×5;
- системные API.

Полный список доступных ресурсов и API зависит от версии СПО с поддержкой EAT.

Разработчик должен знать, что в EAT для локальных переменных выделяется максимум 50 кбайт памяти, при этом размер необходимого объема памяти должен быть предопределен. Это следует учитывать при написании собственных Си-функций.

Минимальный код EAT

Рассмотрим пример минимального пользовательского Си-кода:

```
{
  EatEvent_st event;
  while(1)
  {
      eat_get_event(&event);
      switch(event.event)
      {
          ...
      }
  }
}
```

Здесь ***app_main*** — это точка входа, с которой начинается пользовательская программа, а ***eat_get_event*** — интерфейс для получения событий от ядра ПО модуля.

Таблица 2. Разница распиновок модулей SIM800 и SIM900R

Номер вывода	SIM800	SIM900R
2	GND	не подключен
6	PCM_OUT	не подключен
23	KP_LED	не подключен
24	VBUS	не подключен
27	USB_DP	DBG_TXD
28	USB_DM	DBG_RXD
53	ANT_BT	GND

Пользовательский код выполняется в цикле ***while*** и должен вызывать API-функции, параллельно отслеживая события, которые посылает ядро.

Обычно программисту приходится работать с платой, на которой отдельно стоят GSM-модуль и микроконтроллер (MCU), управляющий GSM-модулем при помощи AT-команд [2]. А в случае с EAT у него возникнет вопрос, как слать GSM-модулю команды и как обрабатывать его ответы. На рис. 3 показан пример, поясняющий, как это можно реализовать в EAT по аналогии с классической архитектурой (раздельный MCU и GSM-модуль).

Видно, что пользовательское ПО по отношению к ядру EAT следует рассматривать как ПО внешнего MCU. Все, что изменится при миграции ПО из внешнего MCU в GSM-модуль, — это синтаксис обращения к ядру EAT и интерпретация результатов. Для пояснения принципа обработки AT-команды ядром приведен пример простейшего кода:

```
void app_main(void)
{
  ...
  Eat_modem_write("AT+CSQ\r",strlen("AT+CSQ\r"));
// Шлем AT-команду
  while(TRUE)
  {
      eat_get_event(&event);
      switch (event.event)
      {
          case EAT_EVENT_MDM_READY_RD: // При
получении ответа от модуля получим индикатор события
EAT_EVENT_MDM_READY_RD
          {
              Progress(); // обрабатываем ответ моду-
ля
          }
          case ...
      }
  }
}
```

Таймеры EAT

Многие программы имеют нелинейный алгоритм исполнения последовательности действий, и для реализации такого поведения программ применяют таймеры, встроенные в MCU. В EAT также реализованы два типа таймеров — программный (***eat_timer_start***) и аппаратный (***eat_gpt_start***) (рис. 4).

Когда программный таймер переполняется, ядро пошлет пользовательскому ПО сообщение «*EAT_EVENT_TIMER*». Пользовательское ПО должно его «отловить» в цикле главной функции ***app_main***. А когда переполнится аппаратный таймер, ядро вызовет callback-функцию. При этом важно, чтобы код внутри callback-функции имел короткий цикл исполнения. Недопустимо, чтобы в ней были заключены блокирующие события, такие как режим сна, ожидание освобождения семафора, работа с памятью и т. п.

Внешние интерфейсы EAT

Рассмотрим примеры работы с некоторыми внешними интерфейсами, такими как GPIO,

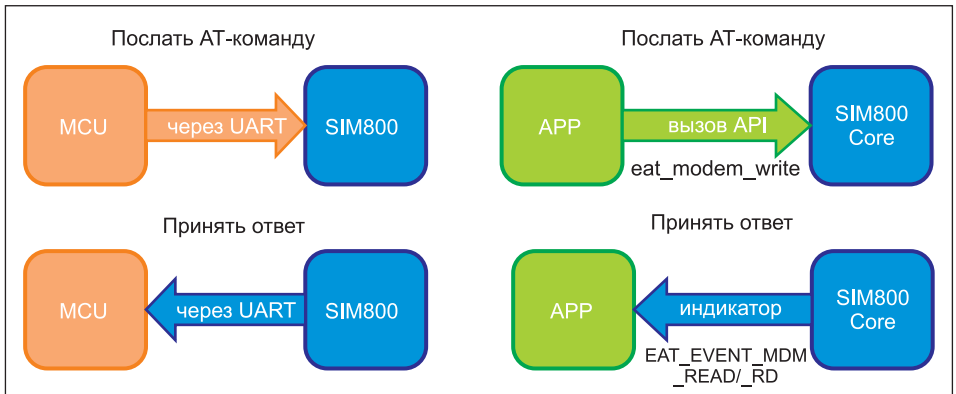


Рис. 3. Принцип работы классической архитектуры (слева) и EAT (справа)

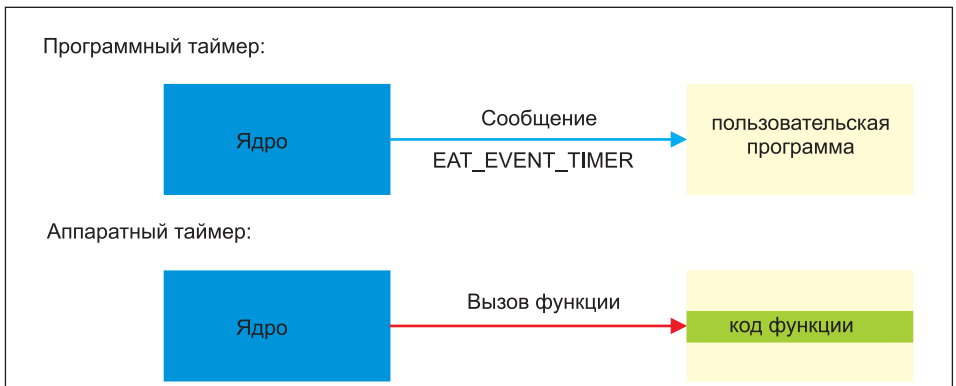


Рис. 4. Пояснение разницы между программным и аппаратным таймерами

Таблица 3. Карта выводов модуля SIM800H и соответствующие функции EAT

Номер вывода	Схемотехническое название вывода	Программное название вывода	Функция в EAT
3	GPIO1	EAT_PIN3_GPIO1	GPIO
4	STATUS	EAT_PIN4_STATUS	GPIO
5	BPI_BUSI	EAT_PIN5_BPI_BUSI	GPIO
20	COL0	EAT_PIN20_COL0	GPIO, KEY_COL
21	COL3	EAT_PIN21_COL3	GPIO, KEY_COL
22	COL2	EAT_PIN22_COL2	GPIO, KEY_COL
23	ROW3	EAT_PIN23_ROW3	GPIO, KEY_ROW
24	COL4	EAT_PIN24_COL4	GPIO, KEY_COL, EINT
25	COL1	EAT_PIN25_COL1	GPIO, KEY_COL
26	PWM	EAT_PIN26_PWM	GPIO, PWM, EINT
27	GPOI2	EAT_PIN27_GPIO2	GPIO, LSA0DA1
28	GPIO3	EAT_PIN28_GPIO3	GPIO, LSCK
29	PCM_CLK	EAT_PIN29_PCM_CLK	GPIO, LSRST, PCMCLK
30	PCM_OUT	EAT_PIN30_PCM_OUT	GPIO, LSDI1, PCMOU
31	RXD	EAT_PIN31_RXD	UART1
32	TXD	EAT_PIN32_TXD	UART1
33	RTS	EAT_PIN33_RTS	GPIO, U1RTS, U2TXD, SPIMISO
34	CTS	EAT_PIN34_CTS	GPIO, U1CTS, U2RXD, SPIMOSI
50	ADC	EAT_PIN50_ADC	ADC
54	SIM_PRESENCE	EAT_PIN54_SIM_PRE	GPIO, EINT, PCMRST
60	ROW1	EAT_PIN60_ROW1	GPIO, KEY_ROW
61	ROW2	EAT_PIN61_ROW2	GPIO, KEY_ROW
62	ROW0	EAT_PIN62_ROW0	GPIO, KEY_ROW
63	ROW4	EAT_PIN63_ROW4	GPIO, KEY_ROW, EINT
64	NETLIGHT	EAT_PIN64_NETLIGHT	GPIO
65	PCM_SYNC	EAT_PIN65_PCM_SYC	GPIO, LSCE0B1, PCMSYNC
66	PCM_IN	EAT_PIN66_PCM_IN	GPIO, LSDA1, PCMIN
68	UART1_RI	EAT_PIN68_UART1_RI	GPIO, U2RTS
69	UART1_DTR	EAT_PIN69_UART1_DTR	GPIO, EINT, PWM
70	UART1_DCD	EAT_PIN70_UART1_DCD	GPIO, U2CTS
74	SCL	EAT_PIN74_SCL	GPIO, SCL28, SPISCK
75	SDA	EAT_PIN75_SDA	GPIO, SDA28, SPICS

SPI и UART (например, в SIM800H). В таблице 3 приведены расположения выводов модуля SIM800H с указанием соответствующих ролей: *PIN No* и *Pin name* — схемотехнический номер и название выводов модуля, *EAT pin name* и ***EAT function*** — программное название вывода и ее функция.

Как видно, многие выводы модуля имеют универсальное назначение. Так, вывод 69 в EAT может быть настроен как GPIO, источник внешнего аппаратного прерывания или вывод PWM. Далее покажем на примерах запись/чтение логических уровней порта и обработку прерываний.

Пример 1 (запись логических уровней):

```
eat_pin_set_mode(EAT_PIN69_UART1_DTR,EAT_PIN_MODE_GPIO); // Настроить вывод 69 как вывод GPIO
eat_gpio_setup(EAT_PIN69_UART1_DTR,EAT_GPIO_DIR_OUTPUT, EAT_GPIO_LEVEL_HIGH); // Вывести на вывод 69 высокий логический уровень
eat_gpio_setup(EAT_PIN69_UART1_DTR,EAT_GPIO_DIR_OUTPUT, EAT_GPIO_LEVEL_LOW); // Вывести на вывод 69 низкий логический уровень
```

Пример 2 (чтение логического уровня):

```
eat_pin_set_mode(EAT_PIN69_UART1_DTR,EAT_PIN_MODE_GPIO); // Настроить вывод 69 как вывод GPIO
eat_gpio_setup(EAT_PIN69_UART1_DTR, EAT_GPIO_DIR_INPUT,0); // Настроить вывод 69 как вход
eat_gpio_read(EAT_PIN69_UART1_DTR); // вернет значение EAT_GPIO_LEVEL_LOW или EAT_GPIO_LEVEL_HIGH
```

Пример 3 (прерывание):

```
void test_handler_int(EatInt_st *interrupt) // объявляем обработчик прерывания
{
  if(interrupt->level)
  {
      //ветка программы, когда уровень на выводе 69 высокий
  }else
  {
      //ветка программы, когда уровень на выводе 69 высокий
  }
}
eat_pin_set_mode(EAT_PIN69_UART1_DTR,EAT_PIN_MODE_EINT); // Настроить вывод 69 как источник прерываний
eat_int_setup(EAT_PIN69_UART1_DTR, EAT_INT_TRIGGER_RISING_EDGE, 10, test_handler_int); // Прерывания по нарастающему фронту с игнорированием эффекта «дребезга» на контакте в 100 мс. При срабатывании прерывания будет вызываться callback-функция test_handler_int.
```

Если вместо callback-функции ввести значение «*NULL*», то ядро будет сообщать о событии «*EAT_EVENT_INT*», которое нужно будет «отловить» в цикле функции ***app_main***.

Выводы 33, 34, 74, 75 могут быть настроены на функцию SPI. Поддерживается: трех- и четырехпроводной SPI; тактовый сигнал 13/26/52 МГц, слова в 8/9/16/24/32 бита. Ниже приведены API-функции для работы с SPI:

- ***eat_spi_init (EAT_SPI_CLK_13M, EAT_SPI_4WIRE,EAT_SPI_BIT8, EAT_FALSE, EAT_TRUE)*** — настройка SPI-интерфейса.

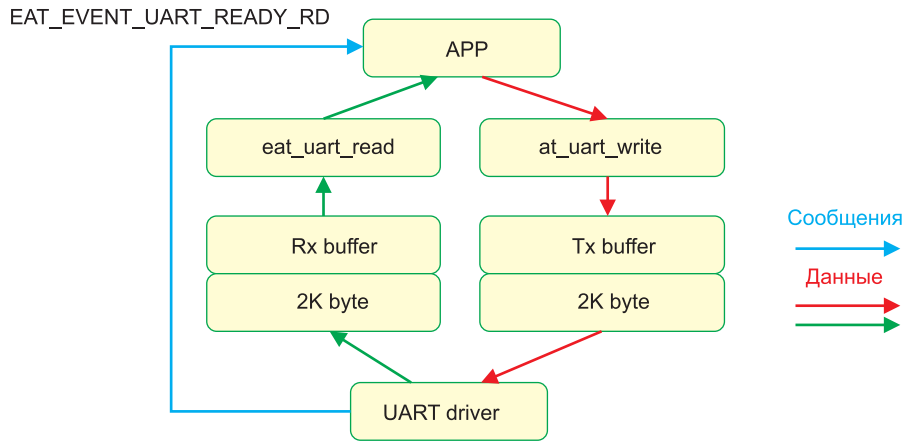


Рис. 5. Принцип работы с портом UART

- ***eat_spi_write (data, len, is_command)*** — запись: <***data***> — указатель на массив данных; <***len***> — длина данных; <***is_command***> — признак C/D («0» — данные, «1» — команда);
- ***eat_spie_read (data, len)*** — чтение: <***data***> — указатель на массив для входящих данных; <***len***> — длина читаемых данных.

Модуль SIM800H поддерживает два порта UART. Оба могут быть настроены в режимы **AT port/Debug port** и **Data mode**. **AT port** предполагает работу с UART как с портом, по которому можно осуществлять управление модулем при помощи AT-команд, как со стандартным GSM-модулем. **Debug port** позволяет снимать диагностическую информацию, которая поможет отследить проблемы, связанные с работой ядра и пользовательского кода. При этом важно учесть, что эти два режима не могут быть назначены UART-порту дважды в одном коде программы. Они должны быть назначены только один раз при инициализации программы.

Data Mode — наверное, самый интересный режим для пользователя, поскольку он дает доступ ко внешним (по отношению к модулю) устройствам. В этом режиме через порт UART можно посылать и принимать данные любого формата. Этот режим не может быть назначен порту, если ранее ему был назначен режим **AT port** или **Debug port**.

Рассмотрим API-функции для работы с UART:

- ***eat_uart_open (EAT_UART_NULL)*** — отменить ранее назначенный режим **AT port** или **Debug port**.
- ***eat_uart_open (EAT_UART_1)*** — открыть порт UART1.
- ***eat_uart_write (EAT_UART_1,buffer, len)*** — послать данные из массива *buffer* длиной *len*.
- ***eat_uart_read (UART,*buffer, len)*** — прочитать данные длиной *len* из UART в массив *buffer*. Функция возвращает реальную длину прочитанных данных.

Для применения этих API-функций нужно понимать принцип работы с UART-портом. Его хорошо поясняет рис. 5. Пользовательское ПО (APP) шлет данные в буфер UART-порта (2 кбит), которые будут высланы вовне из порта UART и ждет (если нужно) ответных данных, приходящих извне в порт UART, отслеживая индикатор события **EAT_EVENT_UART_READY_RD** в цикле функции ***app_main***. Когда индикатор сообщает о приходе данных, они должны быть прочитаны пользовательским ПО из буфера UART.

Прочие API-функции

Выше по тексту был приведен не полный перечень примеров применения API-функций. Актуальный список API-функций и соответствующую документацию для того или иного ПО модуля следует запрашивать у службы технической поддержки [2] или официального представителя компании SIMCom Wireless Solutions. По запросу будут высланы примеры Си-кодов и демопроект Eclipse для интеграции в среду разработки.

Отладочное средство

Рассмотрим, при помощи каких средств и в какой последовательности можно будет скомпилировать код в бинарный файл, записать его в память модуля и отладить в реальности на примере SIM800H.

Для первичного изучения EAT и отладки кода может помочь отладочный набор SIM800H-EVM (рис. 6). В его состав входит все необходимое: основная плата, модуль SIM800H

Рис. 6. Отладочное средство SIM800H-EVM для модуля SIM800H

на мезонине, сетевой адаптер питания, антенна GSM, кабель USB-RS232 с переходником и CD-диском с USB-драйвером, USB-кабель, аудиогарнитура.

Запись ПО в модуль осуществляется посредством UART- или USB-порта. Для этих целей существуют специальные утилиты и драйвер USB для ОС Windows. Более удобным и предпочтительным способом записи ПО в модуль будет применение USB-порта, поскольку UART удобно использовать для отладочных целей или он может быть задействован в целевом приложении, скажем, для работы с каким-нибудь датчиком.

В конечной плате USB-интерфейс должен быть подключен, как показано на рис. 7.

После подключения модуля к порту USB и установки USB-драйвера (поддерживаются OS Windows 98SE/ME/2000/XP/Vista/7) он должен определиться в диспетчере устройств как MTK USB Port (COMXX) (XX — произвольный номер COM-порта). Для работы USB-интерфейса не обязательно запускать модуль сигналом ***POWER KEY***, главное, чтобы на вывод модуля Vbat было подано питающее напряжение.

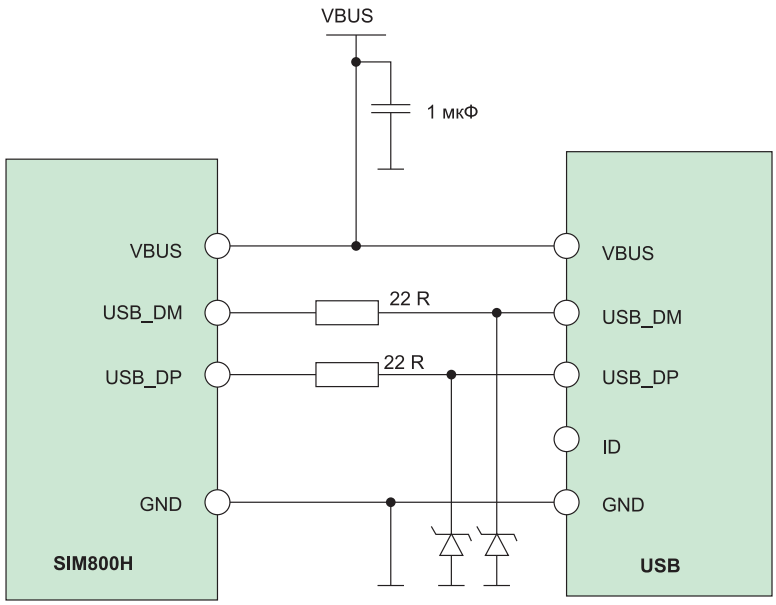


Рис. 7. Схема подключения USB-интерфейса

Программное обеспечение для разработки EAT-приложений

Определившись с аппаратной частью, можно приступать к установке и настройке ПО для разработки EAT-приложений, которое можно запросить у локального дистрибьютора или у службы технической поддержки SIMCom Wireless Solutions [2].

При подготовке данной статьи применялся следующий набор программных продуктов:

- RealView Development Suite v3. 1 со средой разработки Eclipse [1];
- SIM800H_EAT_140516_ECLIPSE, пример проекта под среду Eclipse [2];
- SIMCom_SIM800H_EAT_flash_Tool_V1.01, программа для записи ПО в память программ модуля [2];
- MS_USB_ComPort_Driver_exe_v1.1032, USB-драйвер виртуального COM-порта [2].

После того как среда Eclipse будет установлена, можно воспользоваться готовым примером проекта, импортировав его, как показано на рис. 8а–г.

После нажатия кнопки **Finish** (рис. 8д) нужно очистить проект (рис. 9а, б).

После всего проделанного вы увидите окно проекта (рис. 10). Теперь код примера проекта можно модифицировать и компилировать при помощи интерфейса среды разработки.

Окно среды разработки имеет несколько внутренних окон, имеющих различное назначение. Основным является окно текстового редактора, в котором собственно и пишется Си-код программы. Кстати, надо отметить, что редактор умеет классифицировать текст по содержанию и окрашивать его участки в различные цвета, автоматически выделять начало и конец функций и т. д. Это значительно упрощает процесс написания кода и анализ его текста.

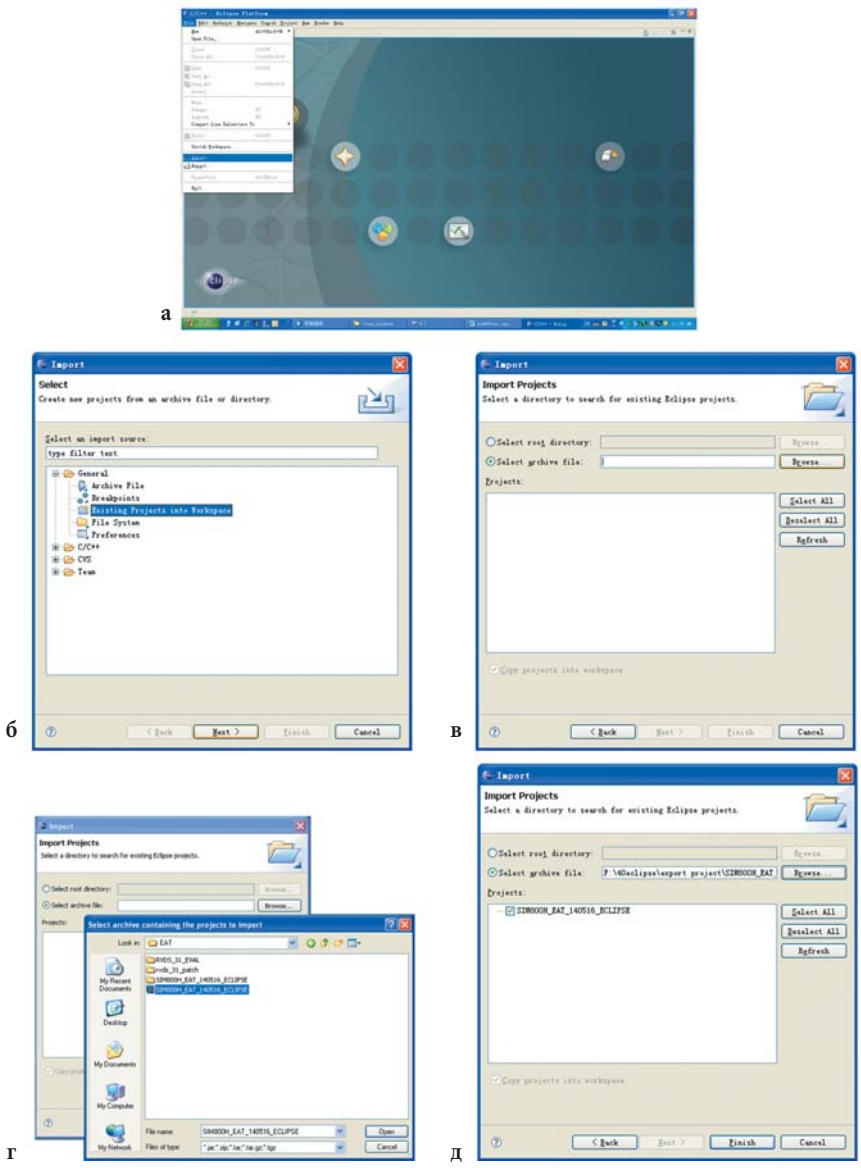
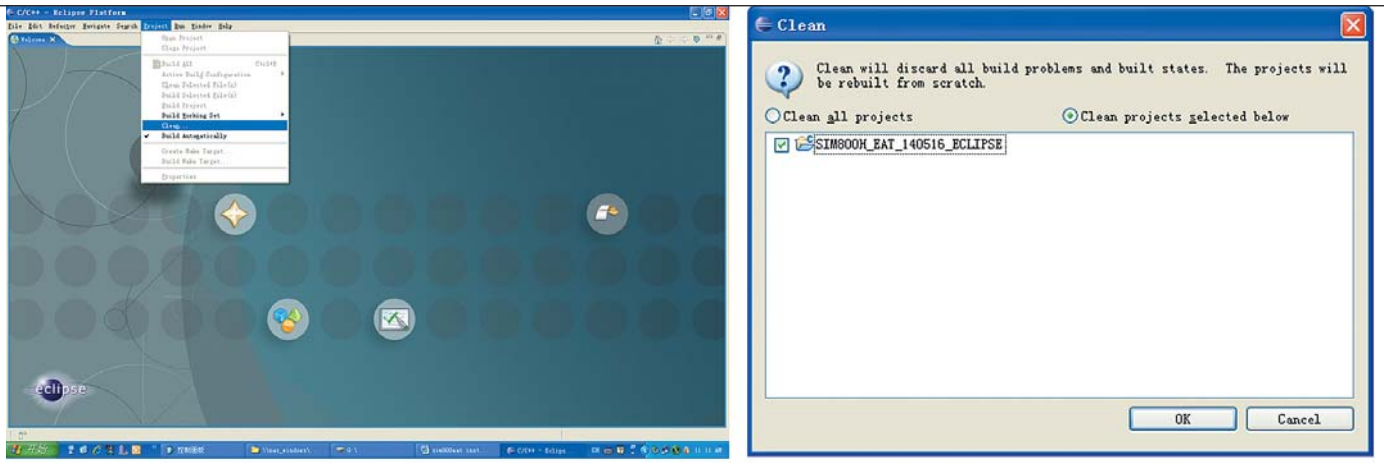


Рис. 8. Импорт примера проекта



а б

Рис. 9. Очистка проекта

Левее от текстового редактора расположен навигатор по проекту, в котором «под рукой» у программиста структура всех файлов, причастных к проекту, включая исходные файлы, бинарный файл ядра, документация, результаты компиляции и проч. Здесь же можно найти руководство программиста с подробным описанием архитектуры EAT и доступных API-функций модуля.

Подробнее о работе Eclipse можно ознакомиться на сайте www.eclipse.org/platform, а мы покажем, как создать свое ПО и записать его в модуль.

Для компиляции кода нужно пройти по меню **Project**->**Build All**, при этом в нижнем консольном окне не должно быть ошибок компиляции, иначе компилятор не создаст результат проекта — бинарный файл EAT, который можно записать в модуль.

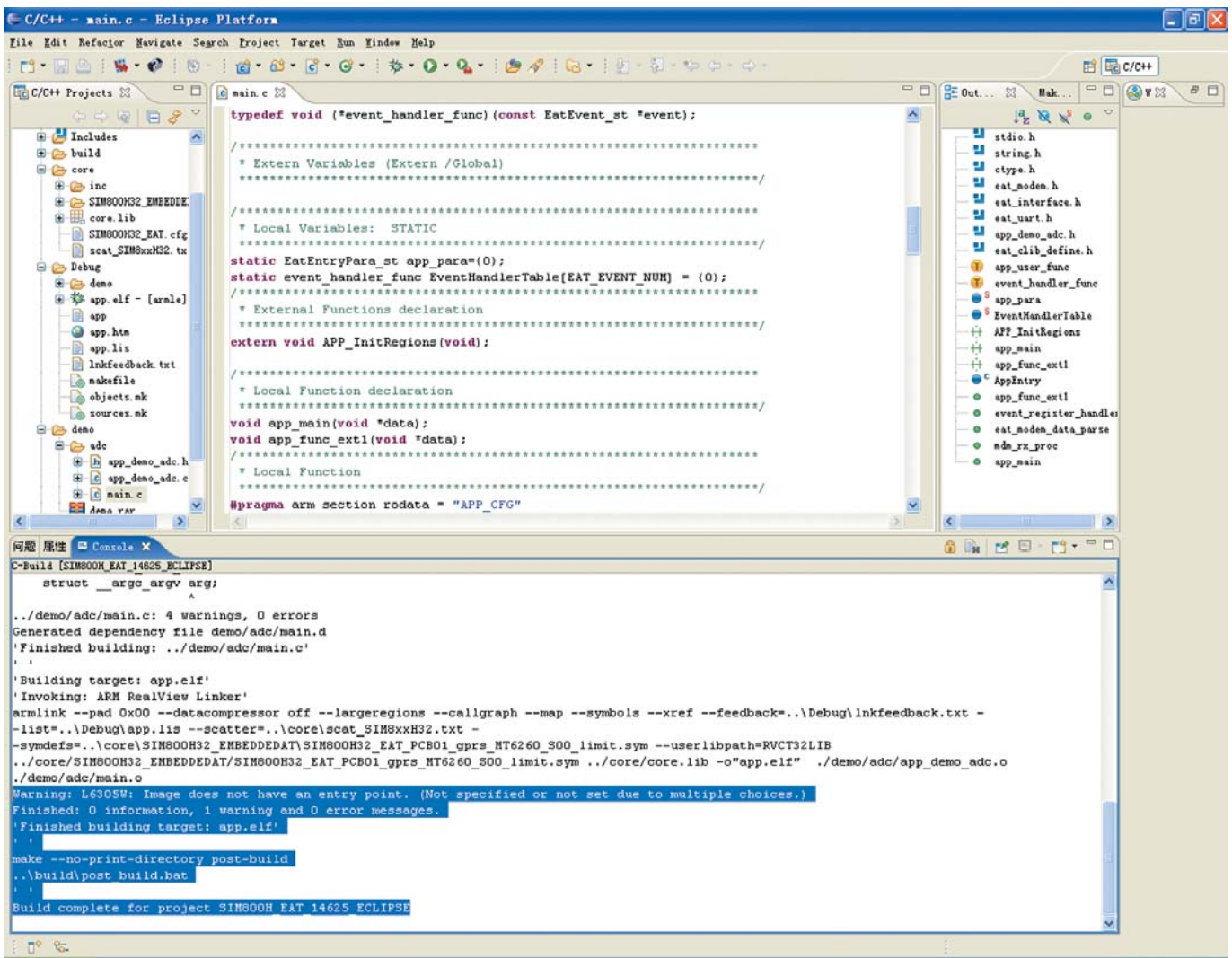


Рис. 10. Окно среды разработки Eclipse с загруженным примером проекта

Запись бинарного файла EAT в память программ модуля

Для загрузки бинарного файла EAT в модуль потребуется программа SIMCom_SIM800H_EAT_flash_Tool (рис. 11).

Модуль SIM800H позволяет загружать ПО через интерфейс USB или UART. В качестве интерфейса для загрузки пользовательского ПО выберем USB. Для этого настроим соответствующим образом программу SIMCom_SIM800H_EAT_flash_Tool.exe, пройдя по меню **Options->USB Download/Readback**. Также потребуется стереть содержимое FAT-модуля, а для этого пройдем по меню **Options->Format FAT (Auto)->Format FAT->OK**.

Укажем путь к пользовательскому ПО, к файлу *SIM800H32_EAT.cfg*, нажав кнопку **Scatter/Config File**. Он должен быть расположен в директории, где расположен проект (указывался при создании проекта), а в данном примере путь такой: *C:\Documents and Settings\Administrator\workspace\SIM800H_EAT_140516_ECLIPSE\core\SIM800H32_EMBEDDEDAT*. Окно программы при этом должно выглядеть так, как на рис. 12.

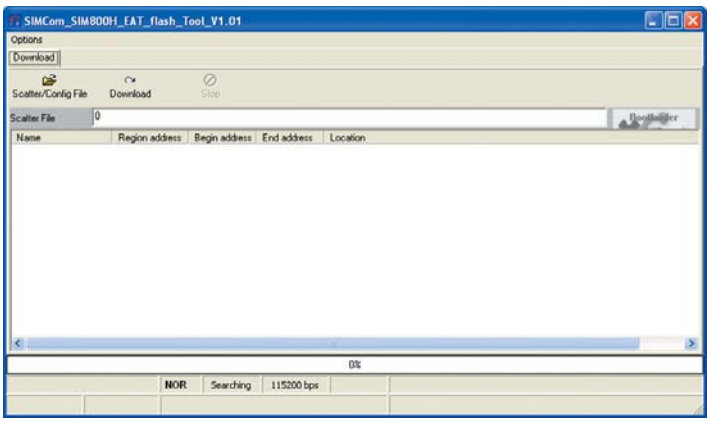


Рис. 11. Окно программы SIMCom_SIM800H_EAT_flash_Tool для загрузки бинарного файла EAT в GSM-модуль

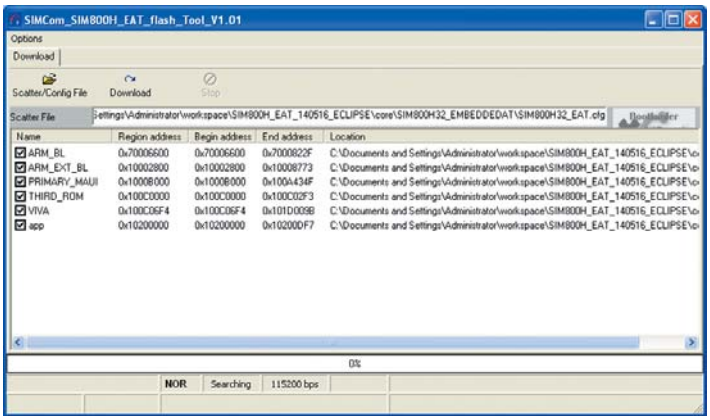


Рис. 12. Окно программы SIMCom_SIM800H_EAT_flash_Tool после настройки

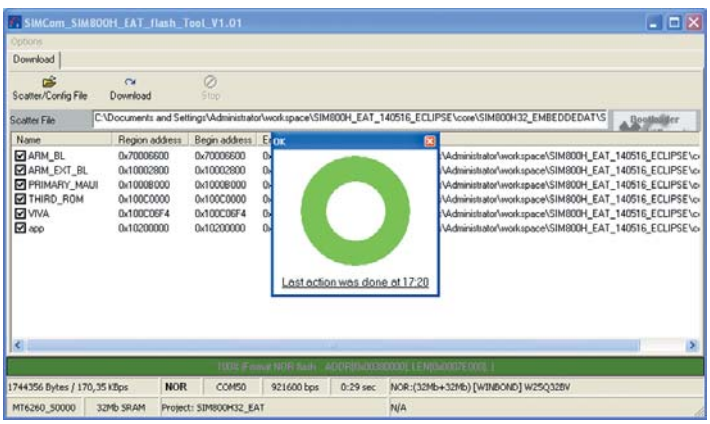


Рис. 13. Окно программы SIMCom_SIM800H_EAT_flash_Tool после успешной загрузки пользовательского ПО в модуль

- Готовим отладочное средство (см. рис. 6), подав питание 5 В от сетевого адаптера, предварительно установив переключатели POWER в состояние «ON», DOWNLOAD — в состояние «OFF».
- Подключаем SIM800H к ПК через кабель USB, при этом потребуется установка USB-драйвера виртуального COM-порта *MS_USB_ComPort_Driver_exe_v1.1032*.
- Устанавливаем переключатель POWER в состояние «OFF», а DOWNLOAD — «ON».
- Нажимаем в окне программы кнопку **Download**.
- Переводим переключатель POWER обратно в состояние «ON», при этом начнется процесс загрузки ПО, по окончании которого появится окно (рис. 13).

Теперь ресурсы модуля SIM800H находятся в полном распоряжении пользовательского кода. После того как пользовательский код будет полностью отлажен и протестирован «в полях», у разработчика возникнет вопрос, как организовать массовое производство устройств с модулем SIM800H. Первой мыслью будет производить загрузку ПО в модуль на производстве после монтажа модулей на плату, но есть и другой вариант. Компания SIMComWireless Solutions предоставляет возможность по предварительной договоренности производить и поставлять через локальных дистрибьюторов модули с загруженным на заводе пользовательским ПО. Такая модель работы возможна при условии подписания соответствующего соглашения о неразглашении, которое в большей степени защищает разработчика ПО от несанкционированного распространения интеллектуальной собственности, что для SIMCom Wireless Solutions является важным аспектом в работе с клиентами по всему миру.

В данной статье было подробно рассказано о технологии Embedded AT, которая открывает для разработчика новые возможности для миниатюризации и удешевления текущих разработок с применением GSM-модулей сотовой связи. В статье не только объясняются теоретические идеи технологии Embedded AT, но и детально на практических примерах показан порядок работы с сопутствующим программным обеспечением для создания пользовательского ПО и его загрузки в GSM-модуль SIM800H. Следуя указаниям, приведенным в данной статье, разработчик сможет максимально быстро начать работу и оценить удобство и функциональность такого решения, как Embedded AT.

Литература

1. https://silver.arm.com/
2. www.sim.com/wm