Разработка мидлетов

для управления аналого-цифровым преобразователем модема G24-J компании Motorola

Статья посвящена методам работы со встроенным аналого-цифровым преобразователем GSM/GPRS-модема G24-J. Рассматриваются примеры мидлетов JavaME для управления внешними и внутренними каналами аналого-цифрового преобразователя. Уделено внимание особенностям использования механизма потоков и средств симуляции аппаратной периферии модема G24-J.

Владимир Каулио, к. т. н.
kv@efo.ru

Использование механизма потоков

Вычислительное ядро модема G24-J представляет собой микроконтроллер ARM7. Наличие одного ядра означает, что в процессе вычислений возможно только последовательное выполнение инструкций процессора. Однако наличие виртуальной машины JavaME позволяет организовать псевдопараллельное выполнение задач, когда временной ресурс разбивается на кванты процессорного времени и распределяется между потоками для выполнения различных задач. В предыдущей статье [1] были рассмотрены программные средства для разработки мидлетов и пример Java-приложения, демонстрирующий использование функций для управления цифровыми линиями ввода/вывода GSM/GPRS-модема G24-J компании Motorola без использования потоков.

В реальных разработках различные функции приложения могут быть реализованы в разных потоках. Это позволяет организовать одновременное выполнение задач и эффективное управление вычислительными ресурсами процессора. Такой подход существенно расширяет функциональные возможности мидлета и упрощает разработку.

В данной статье все примеры реализованы с использованием механизма потоков. Каждый проект состоит из двух файлов: UserMIDlet.java и IO_Thread.java. Первый файл содержит класс UserMIDlet, переменную класса-потока iopin и вызов функции завершения потока iopin.killThread:

```
import javax.microedition.midlet.*;
public class UserMIDlet extends MIDlet
{
   IO_Thread iopin = new IO_Thread();

   public void startApp(){}
   public void pauseApp(){}
   public void destroyApp(boolean unconditional)
   {
         iopin.killThread();
         notifyDestroyed();
   }
}
```

Второй файл отвечает за функциональность приложения, например, содержит класс потока IO_Thread и 3 функции — конструктор, функцию run и функцию завершения потока killThread. Таким образом, достигается локализация функций управления мидлетом от других функций:

```
import com.motorola.oem.hapi.GpioException;
import com.motorola.oem.hapi.GpioInput;
import com.motorola.oem.hapi.GpioOutput;
public class IO_Thread implements Runnable
{
   private boolean running;
   private GpioOutput gpioOutput2;
   private GpioInput gpioInput;
   IO_Thread()
   {
         running = true;
         Thread t = new Thread(this);
         t.start();
   }
   public void run()
   {
         ...
         while(running){...}
   }
   public void killThread()
   {
         running = false;
   }
}
```

В данном случае, в отличие от мидлета в [1], код опроса линии Gpio1 и формирования инвертированного сигнала на линии Gpio2 должен быть помещен в функцию run, которая является стандартной и запускается при создании потока. Выход из функции происходит после завершения цикла while(running). Время выполнения цикла контролируется из основного класса мидлета с помощью переменной-флага running.

Работу программы удобно проверять с помощью симулятора MOTO2MOTO Wireless Toolkit, доступного для скачивания на сайте компании Motorola. Для установки симулятора в среде программирования NetBeans следует открыть

свойства “Properties→Platform→Emulator Platform” и выбрать “MOTO2MOTO Wireless Toolkit 1.2…”. Запуск программы на выполнение — команда “Run Main Project”. В результате на экране появляется окно симулятора OEMDevice (рис. 1), в котором отображается группа состояний линий ввода/вывода Gpio. Зеленым цветом обозначено состояние логической единицы, красным — нуля.

Рис. 1. Окно симулятора OEMDevice версии 1.2

В главном окне также расположена тестовая информация о сети (группа Network), внешних каналах аналого-цифрового преобразователя (группа A2D), состоянии телефона (группа Call), SIM-карте (группа SIM) и системные свойства — режим работы, сторожевой таймер, время/дата, будильник, уровень заряда батареи, температура модуля, присутствие антенны (группа System). Для симуляции внешних воздействий, например, на входе цифровой линии Gpio1 предусмотрено диалоговое окно, вызываемое при нажатии на пункт меню “External Events→ Gpio” (рис. 2). В данном случае уровень сигнала на линии Gpio1 установлен равным 0, при этом согласно программе сигнал на линии Gpio2 инвертировался и стал равен 1.

Рис. 2. Окно симулятора Gpio в симуляторе версии 1.2

Аналого-цифровой преобразователь в составе модема G24-J

Аналого-цифровой преобразователь (АЦП) модема G24-J содержит 3 внешних канала и 2 дополнительных (внутренних), которые предназначены для измерения напряжения питания и температуры модема [2]. Внешние каналы предназначены для измерения постоянного напряжения в диапазоне от 0 до 2,3 В. Разрядность АЦП составляет 10 бит, однако цифровой код внешних каналов содержит только 8 значащих разрядов. Диапазон цифровых целочисленных значений для измеряемых сигналов на внешних входах АЦП — от 0 (0 В) до 255 (2,3 В), для напряжения питания — от 300 до 450 (разрешающая способность 10 мВ), для температуры — от 17 (70 °С) до 229 (–30 °С).

Каждый из пяти каналов АЦП может работать в одном из трех режимов:

- Однократное измерение: преобразование выполняется по запросу.
- Автоматическое периодическое измерение: АЦП производит измерения с частотой, установленной в специальном регистре. Уведомление о результате выполнения каждого измерения реализовано с помощью механизма сообщений.
- Автоматическое периодическое измерение с предустановленным пределом: АЦП производит измерения с заданной частотой, пользователь определяет верхнее и нижнее пороговые значения. Измеренное значение сравнивается с пороговыми значениями, и сообщение генерируется только в том случае, если пороговые значения превышены.

В модеме G24-J управление внешними и внутренними каналами реализовано по-разному. Рассмотрим вариант управления переключением состояния цифрового выхода в зависимости от состояния цифрового входа (вкл/выкл) и результата измерения на первом канале АЦП.

Для использования внешних каналов АЦП необходимо получить доступ к функциям классов A2dManager, A2dChannel и разрешить обработку исключений при работе с АЦП — A2dException. Первый класс необходим для получения доступа к выбранному каналу, второй — для считывания результатов аналого-цифрового преобразования.

Существенное отличие использования внешних аналоговых линий АЦП от цифровых линий состоит в том, что для создания объекта класса A2dManager следует применять функцию getInstance. Функция возвращает ссылку на объект A2DChannel исходя из номера канала (от 1 до 3). Класс A2DChannel содержит методы для конфигурации и получения уведомлений от внешних аналоговых каналов. В частности, метод getValue возвращает результат аналого-цифрового преобразования. Данная функция блокирует выполнение потока, из которого она была вызвана, до тех пор, пока не завершится ее выполнение. Такой эффект легко заметить и при работе с симулятором.

Модифицируем алгоритм инвертирования цифровых входов следующим образом. Если значение АЦП на первом канале меньше 1,13 В, то выход Gpio2 равен инвертированному значению Gpio1. В противном случае, выход Gpio2 повторяет значение на входе Gpio1:

```
import com.motorola.oem.hapi.GpioException;
import com.motorola.oem.hapi.GpioInput;
import com.motorola.oem.hapi.GpioOutput;
import com.motorola.oem.hapi.A2dException;
import com.motorola.oem.hapi.A2dManager;
import com.motorola.oem.hapi.A2dChannel;
public class IO_Thread implements Runnable
{
    private boolean running;
    private GpioOutput gpioOutput2;
    private GpioInput gpioInput;
    private A2dManager adc;
    private A2dChannel channel;
    private int adc_value;
    private boolean flag;
    private A2dManager adc;
    private A2dChannel channel;
    private int adc_value;
    private boolean flag;
    IO_Thread()
    {
        running = true;
        Thread t = new Thread(this);
        t.start();
    }
    public void run()
    {
        try
        {
            gpioOutput2 = new GpioOutput(2);
            gpioInput = new GpioInput(1);
            adc = A2dManager.getInstance();
        }
        catch (GpioException e)
        {
            if(gpioOutput2 != null)
                gpioOutput2.clear();
            if(gpioInput != null)
                gpioInput.clear();
            e.printStackTrace();
        }
        catch (A2dException e){;}
        while(running)
        {
            try
            {
                channel =
                adc.getChannel(A2dManager.CH1);
                adc_value = channel.getValue();
                if(adc_value < 125)
                    flag = false;
                else
                    flag = true;
                if(gpioInput.read())
                    gpioOutput2.write(flag);
                else
                    gpioOutput2.write(!flag);
                Thread.sleep(100);
            }
            catch (GpioException e){;}
            catch (A2dException e){;}
            catch (InterruptedException e){;}
        }
    }
    public void killThread()
    {
        running = false;
    }
}
```

Результат работы данного мидлета показан на рис. 3. С помощью диалогового окна A2D устанавливается желаемое значение входной измеряемой величины на первом канале АЦП. При значении, равном 0,5 В, индикаторы цифровых линий Gpio1 и Gpio2 в окне OEMDevice имеют разные цвета (красный и зеленый), при значении 1,13 В — цвет индикаторов одинаковый. Данную программу также удобно проверять на отладочном комплекте [3]. Для этого необходимо задействовать управляющие переключатели из группы Gpio и ADC, расположенные слева и справа от разъема SIM-карты.

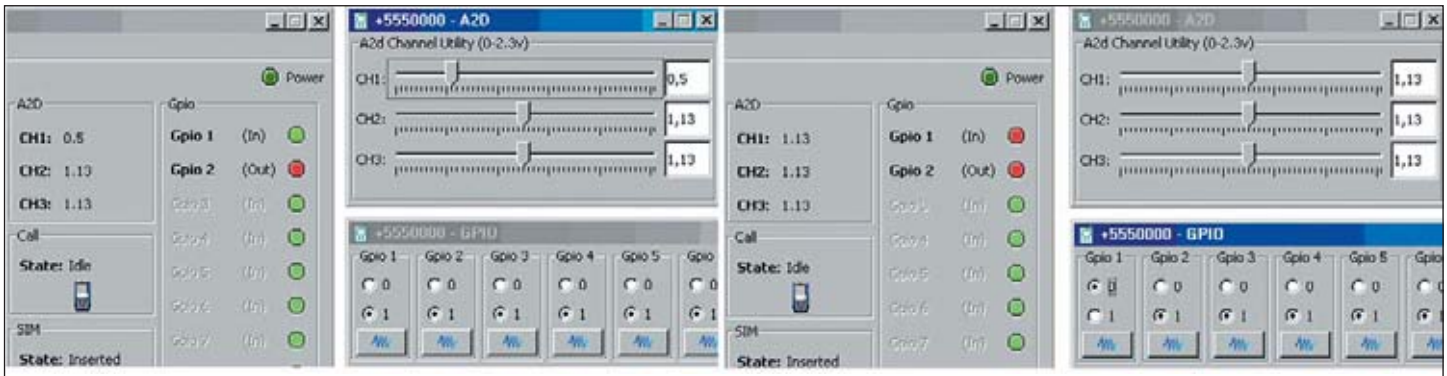

Рис. 3. Демонстрация использования АЦП и Gpio в модеме G24-J

Внутренние каналы аналого-цифрового преобразователя G24-J

Внутренние каналы АЦП предназначены для анализа напряжения питания модема G24-J и слежения за его температурой. Функции слежения за питанием и температурой расположены в пакете OSC. Этот пакет включает также будильник, дату и время, антенну (наличие подключения), сторожевой таймер и др. В программу можно добавить возможность приема уведомлений при изменении напряжения питания, либо осуществлять регулярный опрос с помощью функции OSC.getBatteryLevel.

Рассмотрим вариант с автоматическим уведомлением на примере индикации уровня заряда батареи. Определим несколько дискретных уровней заряда и будем использовать цифровые выходы для включения/выключения светодиодов. Данный подход не является наилучшим решением с точки зрения энергопотребления, однако позволяет наглядно продемонстрировать результаты работы.

В определение класса потока добавляется спецификатор OSCBatteryListener. В инициализирующей части функции run указывается механизм уведомлений OSC.setBatteryListener (this), размещается код для его включения OSC.batteryLevelChangeReportEnable(true), а также настраиваются цифровые выходы. Функция, отвечающая за обработку уведомлений, называется onBatteryLevelChanged. В ней реализуется обработка уведомления об изменении напряжения питания. В данном случае определяется уровень, к которому следует отнести текущее значение напряжения, и устанавливаются соответствующие сигналы на цифровых выходах.

Управляющий элемент симулятора позволяет задать значения напряжения в диапазоне от 3,2 до 4,2 В. Определим минимальный уровень напряжения, при котором отображается уровень заряда, равный 3400 (3,4 В), максимальный — от 4000 (4,0 В).

Анализ функции контроля напряжения питания удобнее выполнять в симуляторе, поскольку имеется возможность регулировки в режиме on-line. На рис. 4 показан процесс изменения напряжения питания с индикацией. При значении заряда батареи, равном 3,3 В, светодиодные индикаторы, подключенные к линиям ввода/вывода Gpio3/4/5/6, имеют красный цвет, то есть значение равно 0. По мере нарастания напряжения сначала срабатывает индикатор Gpio6 (меняет цвет с красного на зеленый), затем добавляются Gpio5 и Gpio4. Согласно программе, при достижении уровня 4 В на всех линиях Gpio3/4/5/6 установится значение 1. Таким образом, реализован альтернативный вариант демонстрации степени заряда батареи по сравнению со встроенным в симуляторе индикатором Battery Level.

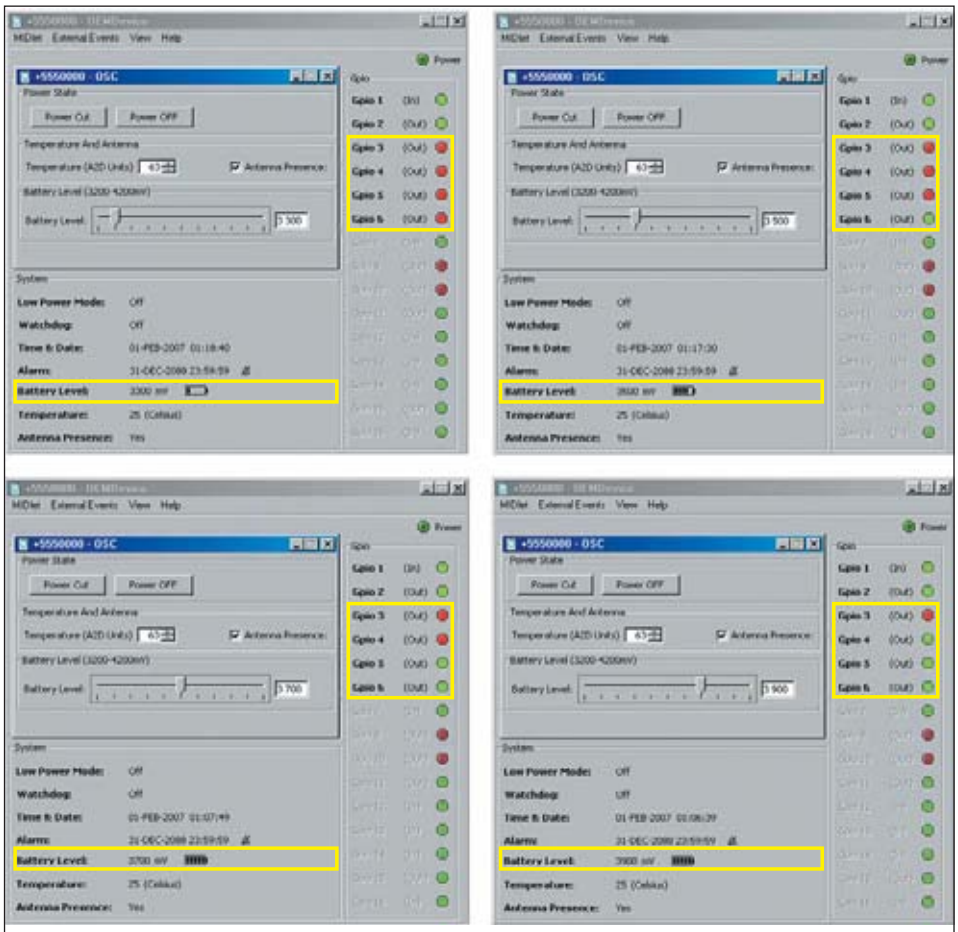

Рис. 4. Демонстрация использования внутреннего канала АЦП

```
public void onBatteryLevelChanged(int voltage)
{
  try
  {
    if(voltage < 3400)
    {    // low voltage
         gpioOutput3.write(false);
         gpioOutput4.write(false);
         gpioOutput5.write(false);
         gpioOutput6.write(false);
    }
    else if(voltage >= 3400 && voltage < 3600)
    {    // level 1
         gpioOutput3.write(false);
         gpioOutput4.write(false);
         gpioOutput5.write(false);
         gpioOutput6.write(true);
    }
    else if(voltage >= 3600 && voltage < 3800)
    {    // level 2
         gpioOutput3.write(false);
         gpioOutput4.write(false);
         gpioOutput5.write(true);
         gpioOutput6.write(true);
    }
```

```
    else if(voltage >= 3800 && voltage < 4000)
    {     // level 3
          gpioOutput3.write(false);
          gpioOutput4.write(true);
          gpioOutput5.write(true);
          gpioOutput6.write(true);
    }
    else if(voltage >= 4000 && voltage <= 4700)
    {     // level 4
          gpioOutput3.write(true);
          gpioOutput4.write(true);
          gpioOutput5.write(true);
          gpioOutput6.write(true);
    }
    else {;} // overvoltage
}
catch (GpioException e){;}
}
```

Заключение

В статье рассмотрены примеры мидлетов для работы с аналого-цифровым преобразователем GSM/GPRS-модема G24-J компании Motorola. Рассмотрен механизм организации потоков для считывания результатов измерения с внешних каналов аналого-цифрового преобразователя, а также механизм уведомлений для анализа изменений напряжения питания.

Приведенные примеры наглядно демонстрируют удобство использования средств JavaME для работы с периферией GSM/GPRS-модема G24-J. Наличие такого мощного средства управления наряду с возможностью обеспечения беспроводной сотовой связи позволяет реализовывать компактные системы сбора и передачи данных. В ряде случаев наличие 3 аналоговых и 15 цифровых линий позволяет создавать беспроводные системы без использования внешнего микроконтроллера. Таким образом, модем G24-J — одно из перспективных решений в области современных встраиваемых средств сотовой связи.

Литература

1. Каулио В. В. Методы и средства разработки мидлетов для GSM-модемов G24-J компании Motorola // Беспроводные технологии. 2008. № 3.
2. Motorola G24 Developer's Guide. Module Hardware Description. Technical information. December 31, 2007.
3. Motorola G24 Developer's Guide. Developer's Kit. Technical information. May, 2008.