Преобразователь интерфейса I²C <-> UART на основе микроконтроллера UPD78F9222 (NEC)

Сергей КОРЮКИН
sergey.koryukin@eltech.ur.ru

Нередко возникает необходимость в различных преобразователях интерфейсов. В данной статье хотелось бы рассмотреть реализацию преобразователя последовательного интерфейса I²C в UART. В качестве преобразователя будет использоваться недорогой микроконтроллер фирмы NEC — μPD78F9222, принадлежащий к семейству 78K0S/KA1+ [1]. Микроконтроллер имеет Flash объемом 4 кбайт, 2 кбайт ОЗУ, встроенный интерфейс UART, программируемый супервизор питания, встроенный узел сброса, 8- и 16-разрядные таймеры, встроенные тактовые генераторы: один для ядра, другой для сторожевого таймера, систему прерываний и прочее. Весьма привлекательна и малая стоимость устройства. Однако интерфейс I²C необходимо эмулировать программно.

При работе с шиной I²C микроконтроллер (МК)будет выступать в роли «ведущего» (I²C-Master). В качестве «ведомого» (I²C-Slave) рассматривается цифровой датчик **ADT75** фирмы Analog Devices. Это 12-разрядный датчик температуры с рабочим диапазоном температур от –55 до +125 °С, погрешностью измерения ±2 °С и интерфейсом I²C. Одновременно к шине I²C может быть подключено до 8 подобных датчиков (всего в сети I²C допускается до 128 устройств).

Для создания прототипа преобразователя использовался демонстрационный набор от NEC «**Low Pin Count — Do it!**» [2] с установленным микроконтроллером **uPD78F9222** (рис. 1). Демонстрационная плата подключается к компьютеру через USB-интерфейс, при этом на хост-компьютере создается виртуальный COM-порт, через который происходит как программирование МК, так и обмен данными между МК и хост-компьютером.

В состав набора входит несколько демонстрационных программ, позволяющих ближе познакомиться с МК данного семейства. Программирование МК осуществляется при помощи входящей в состав комплекта утилиты.

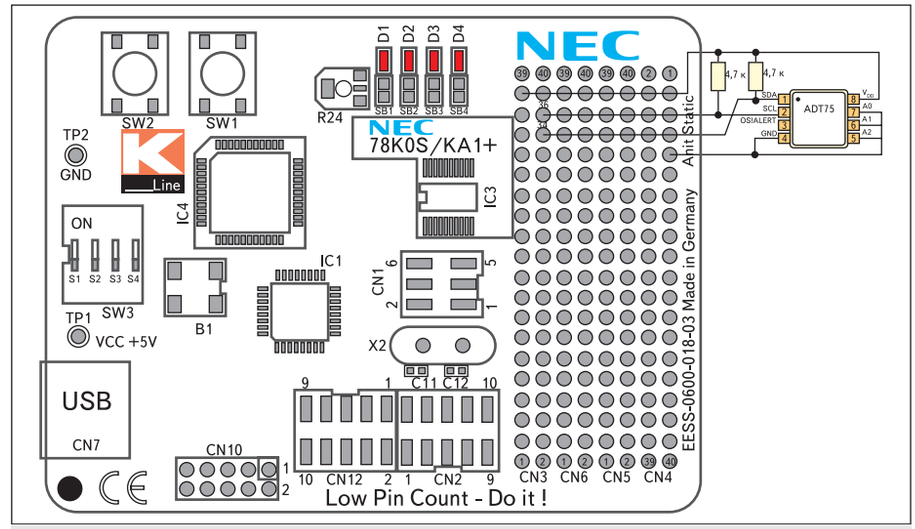

Рис. 2. Схема соединений

Среди представленных демонстрационных программ есть ADC Demo, на ее примере обеспечивается интерактивное управление оценочным комплектом через гипертерминал персонального компьютера. Эта программа была взята за основу программной эмуляции интерфейса I²C-UART. Рассмотрим подробно варианты решения данной задачи.

Рис. 1. Набор «Low Pin Count — Do it!»

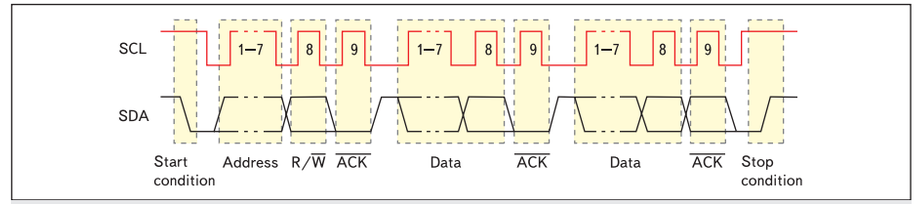

Рис. 3. Временная диаграмма обмена данными интерфейса I²C

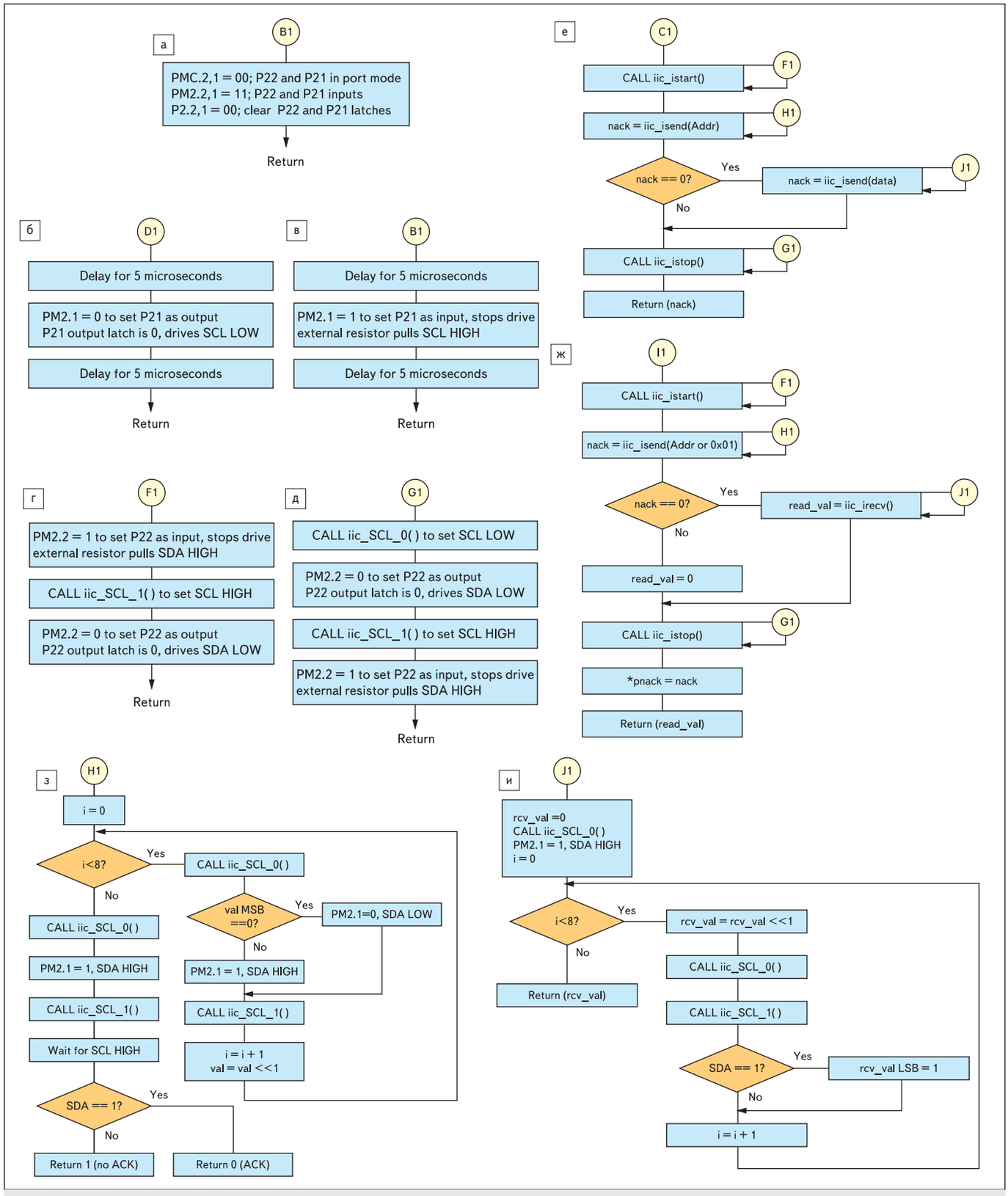

Рис. 4. Блок-схемы процедур:
а) iic_init() — инициализация портов; б) iic_SCL_0()- сброс SCL в «0»; в) iic_SCL_1() — установка SCL в «1»; г) iic_istart() — формирование сигнала Start condition (начало передачи); д) iic_istop() — формирование сигнала Stop condition (конец передачи); е) iic_dkwr(addr, data) — запись байта в Slave; ж) iic_dkrd(addr, *pnack) — чтение байта из Slave; з) iic_isend(val) — поразрядная передача байта в порт P22; и) iic_irecv() — поразрядное чтение байта из порта P22

На рис. 2 приведена схема подключения датчика к плате оценочного комплекта.

Контактные площадки 34 и 36 макетного поля CN3 подключены к выводам 17 и 18 микроконтроллера IC3 и являются портами P22 и P21 соответственно. Порт P22 отвечает за

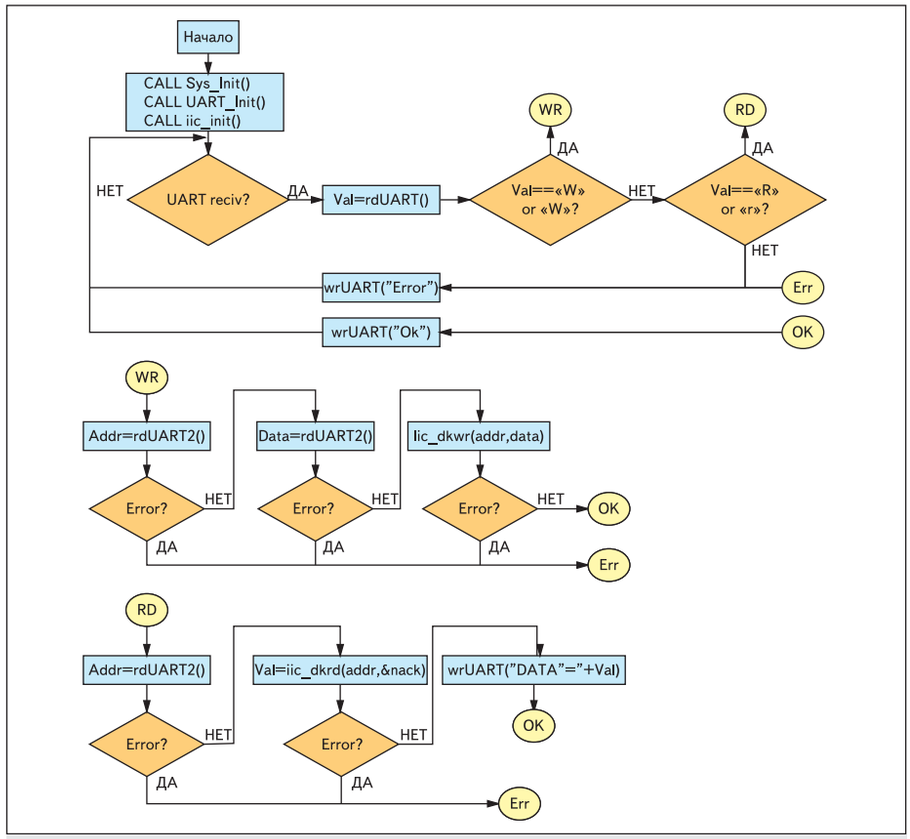

Рис. 5. Блок-схема программы

сигнал SDA, а P21 — за SCL шины I²C. Временная диаграмма обмена данными через интерфейс I²C изображена на рис. 3.

Для обеспечения работы интерфейса I²C необходимы следующие процедуры:

Процедуры нижнего уровня:

- **iic_init()** — инициализация портов;
- **iic_SCL_0()** — сброс SCL в «0»;
- **iic_SCL_1()** — установка SCL в «1»;
- **iic_istart()** — формирование сигнала Start condition (начало передачи);
- **iic_istop()** — формирование сигнала Stop condition (конец передачи).

Процедуры верхнего уровня:

- **iic_dkwr(addr, data)** — запись байта в Slave (ADT75);
- **iic_isend(val)** — поразрядная передача байта в порт P22;
- **iic_dkrd(addr, *pnack)** — чтение байта из Slave (ADT75);
- **iic_irecv()** — поразрядное чтение байта из порта P22.

На рис. 4 представлены блок-схемы этих процедур с кратким описанием.

Задержки длительностью 5 мкс в процедурах **iic_SCL_0()** и **iic_SCL_1()** необходимы для организации скорости обмена в 100 кбит/с.

В процедурах **iic_dkwr(addr, data)** и **iic_dkrd(addr, *pnack)** **addr** — адрес внешнего устройства на шине I²C (значение от 0 до 127, сдвинутое на 1 бит влево), data — данные. В процедуре чтения **iic_dkrd(addr, *pnack)** при вызове процедуры **iic_isend(val)** младший бит addr устанавливается в «1», указывая на то, что происходит чтение из устройства.

Основной цикл программы представлен на диаграмме рис. 5. Изначально выполняются процедуры инициализации МК. Затем программа зацикливается, ожидая прерывания от интерфейса UART, которое поступает при приеме 1-го байта из хост-компьютера. Далее идет анализ принятого байта и, в зависимости от принятого символа, следует переход к записи информации в Slave (WR, если принят символ W или w) или к чтению байта из Slave (RD, если принят символ R или r). Если же ни одно условие не выполняется, следует переход на процедуру передачи хосту сообщения об ошибке — Error.

При записи байта в Slave требуются дополнительные данные от компьютера: адрес устройства (Addr) и непосредственно данные (Data). Эти данные считываются процедурой rdUART2(), которая также преобразует принятые 2 символа в десятичный формат. Затем управление передается процедуре выдачи данных на шину I²C — iic_dkwr(Addr, Data). Формат данных, передаваемых из компьютера, должен быть в виде Waadd, где W — команда записи, aa — адрес устройства умноженный на 2 в шестнадцатеричном представлении (например: 4Ah = 74 = 37×2, где 37 — десятичный адрес устройства), dd — сами данные для записи в шестнадцатиричном виде (например: A8h = 168). Пример команды — w4AA8. Если при выполнении команды записи происходит ошибка, то в компьютер передается соответствующее сообщение, иначе передается сообщение Ok.

При чтении байта из Slave требуется только адрес устройства, представленный в том же виде, что и в команде записи. Форма команды имеет вид: Raa, где R — команда на чтение, aa — адрес устройства. Если все параметры заданы верно, то в UART передается сообщение DATA=, считанный из устройства байт данных и сообщение Ok. Если же возникли ошибки, то передается сообщение Error.

После окончания процедуры записи или считывания, а также если были ошибки, МК переходит в режим ожидания приема байта команды.

Несколько слов об используемых процедурах:

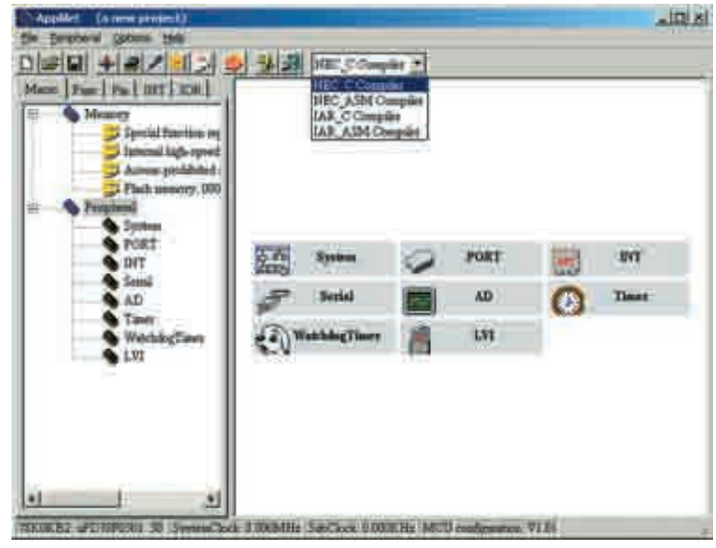

Рис. 6. Бесплатная среда настройки периферии МК NEC

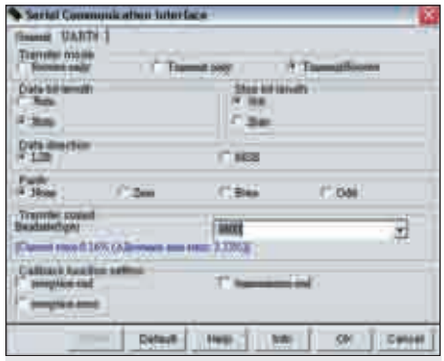

Рис. 7. Настройка UART в Applilet

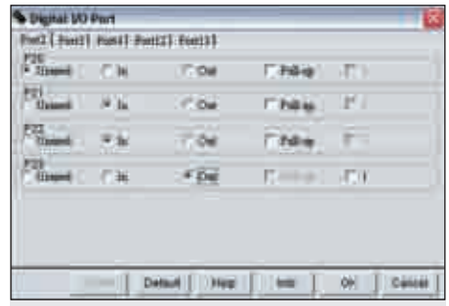

Рис. 8. Настройка портов P21 и P22

- rdUART() — прием одного байта данных из UART;
- rdUART2() — прием двух байтов из UART и преобразование их в десятичный формат;
- wrUART() — передача сообщений в UART.

Для облегчения настройки периферии микроконтроллеров NEC у разработчика есть возможность использовать бесплатную среду Applilet (рис. 6).

Эта среда позволяет за считанные минуты сформировать исходный код, необходимый для инициализации периферийных устройств используемого МК, на языке С или ассемблера. Все действия сводятся к установке нужных галочек и ввода необходимых параметров. На рис. 7 и 8 приведен пример настройки UART и портов P21, P22, используемых в настоящем проекте для формирования сигналов интерфейса I²C.

Входящий в оценочный комплект софт от IAR включает:

- менеджер проектов (интегрированная среда разработки);
- ассемблер;
- Си-компилятор;
- симулятор.

Этот софт позволяет создавать полноценные программные проекты для микроконтроллеров **uPD78F9222**. После того как программа откомпилирована и загружена в МК оценочного комплекта, можно запустить стандартную программу HyperTerminal, входящую в ОС Windows, настроить ее на работу с виртуальным COM-портом, выставить параметры связи — такие же, как и в настройках UART-микроконтроллера. Теперь хост-компьютер играет роль терминала для оценочного комплекта.

После сброса МК на терминале должна появиться надпись: I²C <-> UART. Можно начинать процедуры обмена данными. На рис. 9 приведен пример считывания текущей температуры из ADT75. Посылаем команду W9000 — адрес устройства 48h (умножаем на 2, получаем 90h, см. DataSheet на ADT75 [3]) и адрес регистра температуры 00. После получения подтверждения выполнения команды Ok, посылаем команду считывания из устройства 90 — R90. Получаем содержимое регистра температуры, в нашем случае оно равно 19h, что соответствует 25 °С (судя по описанию на ADT75).

Конечно, программу можно доработать и посылать в компьютер уже вычисленное значение температуры. Причем можно немного усложнить и считывать 2 и более байта данных с шины I²C. Так как м/с датчика температуры имеет разрядность 12 бит, мы считываем только старшие 8, но это уже тема для другой статьи.

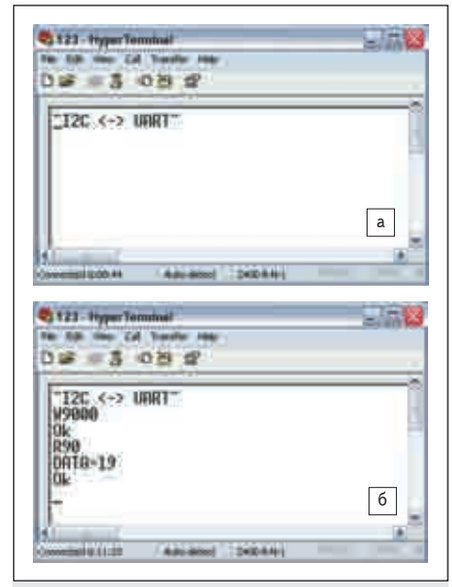

Рис. 9. Пример обмена данными

В завершение хочется напомнить, что отладочным комплектом «**Low Pin Count — Do it!**» можно воспользоваться как программатором. Достаточно вывести линии X1_cpld, X2_cpld, RES_N, VCC_uPD и GND с платы на отдельный разъем и подключать его к собственной разработке.

Один из вариантов схемы готового устройства приведен на рис. 10. «Лишние» узлы можно смело выкидывать (за исключением МК). Исходные коды полученного проекта можно найти на сайте [4].

Литература

1. ww w.eltech.spb.r u/pdf/nec/234/nec_234.pdf
2. w ww.eltech.spb.r u/techinfo.ht ml?aid=243
3. w ww.analog.c om/en/prod/0,2877,ADT75,00.ht ml
4. w ww.eltech.spb.r u/addons/NEC_I2C-UART.rar
5. w ww.semtech.c om/products/product-detail.jsp?navId=H0,C157,C159,P1471

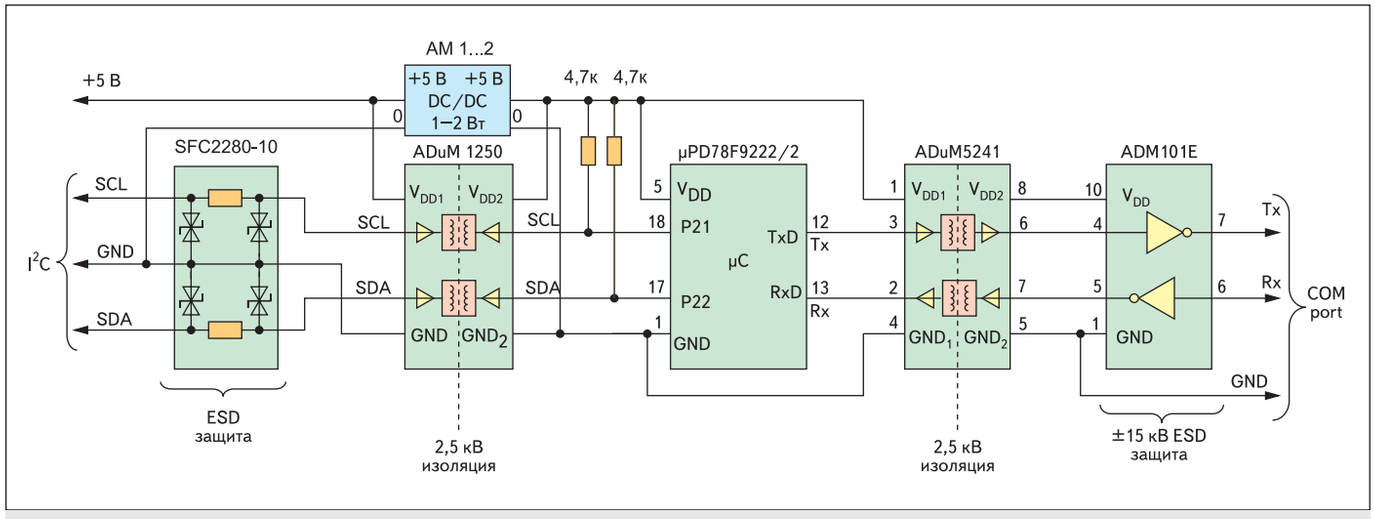

Рис. 10. Пример реализации преобразователя I²C <-> UART