Универсальный кабель для внутрисхемного программирования

Традиционный вопрос, встающий перед каждым, кто начинает работать с новым для себя типом микроконтроллеров или микросхем программируемой логики — «а как их программировать?» К счастью, постепенно забываются те времена, когда для программирования микроконтроллеров приходилось покупать достаточно дорогие программаторы, и все больше типов современных микроконтроллеров допускают внутрисхемное программирования с использованием простых кабелей, подключаемых к параллельному порту компьютера.

Кроме того, в наиболее продвинутых микроконтроллерах присутствует возможность отладки через JTAG-порт (как это сделано, например, в микроконтроллерах семейства MSP430 фирмы Texas Instruments). И все было бы совсем хорошо, если бы разные типы микроконтроллеров можно было программировать одним единственным кабелем. Ведь все эти кабели для внутрисхемного программирования представляют собой просто шинный формирователь, включенный между параллельным портом компьютера и программируемым устройством. В большинстве случаев можно обойтись даже и без него, просто соединив соответствующие выводы LPT-порта и микроконтроллера, однако делать это не рекомендуется, поскольку обычно используется достаточно длинный кабель, а многие микроконтроллеры чувствительны к чистоте тактового сигнала при программировании.

В настоящее время автору данной статьи известны следующие наиболее распространенные типы кабелей для внутрисхемного программирования микроконтроллеров и микросхем программируемой логики:

- кабели типа STK200/STK300 фирмы Atmel для программирования микроконтроллеров семейства AVR; с кабелями этого типа работает программное обеспечение фирмы Atmel, а также замечательная программа AVReal, написанная Александром Редчуком;
- кабели типа ByteBlaster/ByteBlasterMV фирмы Altera, предназначенные для начального программирования ПЛИС практически всех семейств этой фирмы; кроме того, этот кабель может использоваться и для

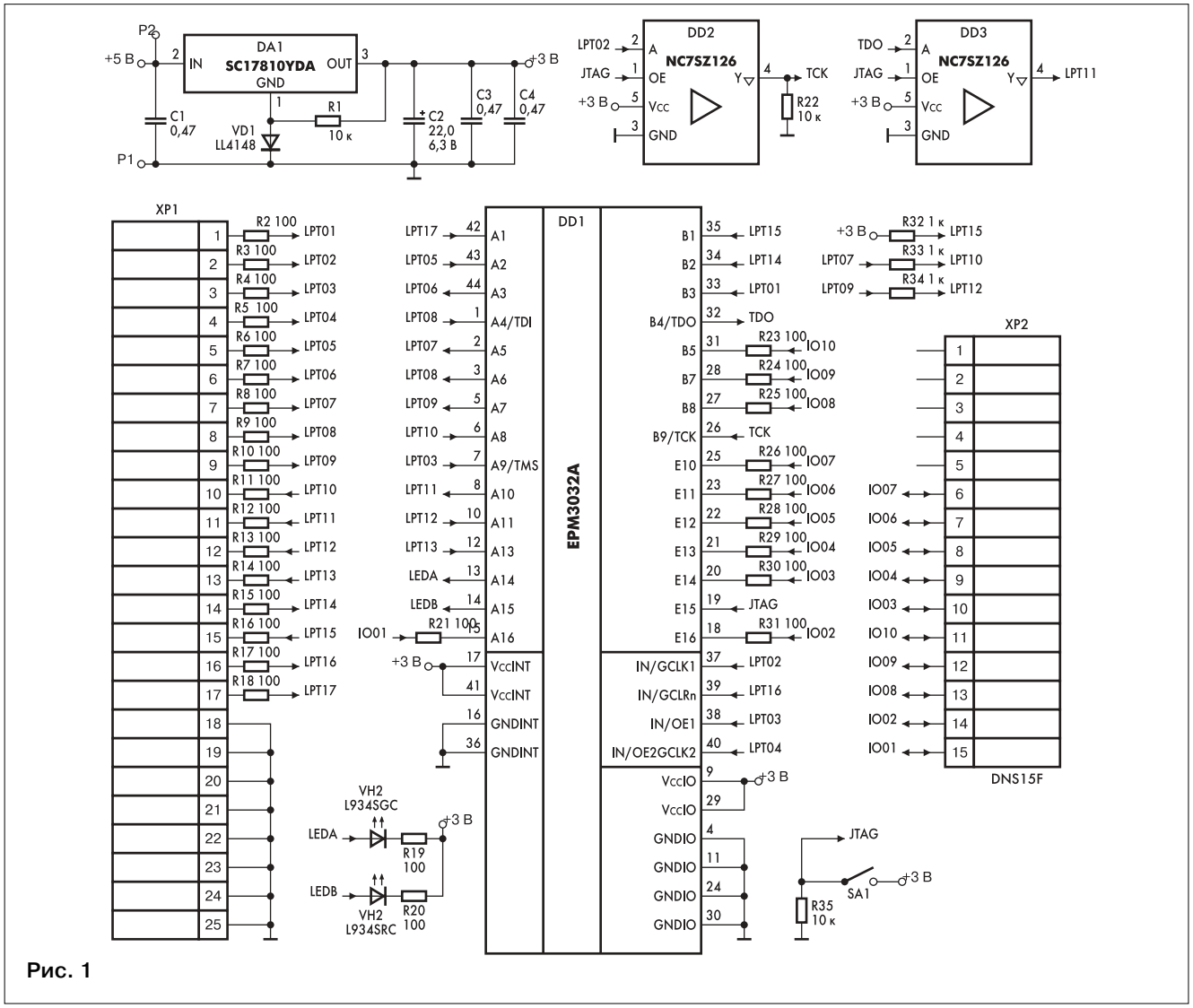

Рис. 1

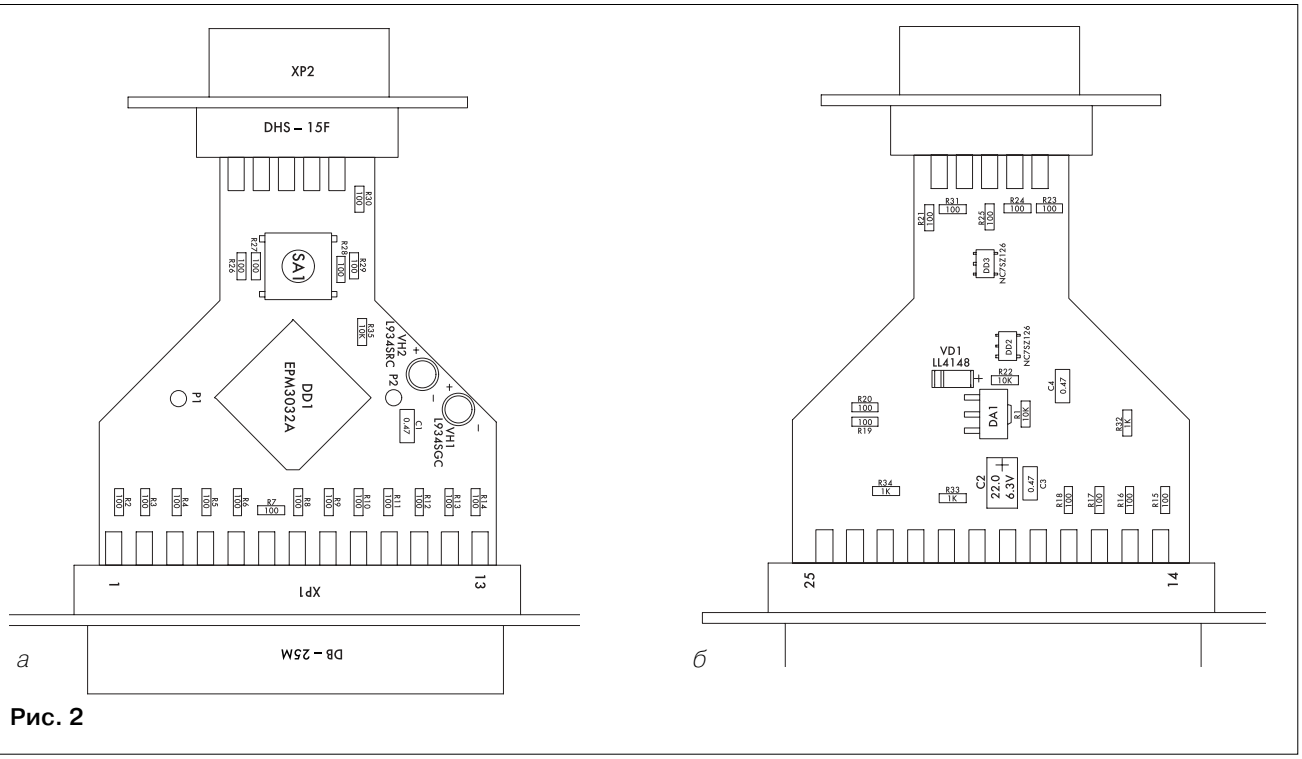

Рис. 2

программирования микро-контроллеров семейства AVR, если делать это с помощью программы AVReal;

- кабель JTAG Parallel Download Cable фирмы Xilinx — для программирования и начальной инициализации ПЛИС этой фирмы;
- JTAG-кабель для программирования микроконтроллеров семейства MSP430 фирмы Texas Instruments; при использовании этого кабеля, помимо программирования, возможна также отладка программ на работающем кристалле прямо в системе с использованием встроенных возможностей этих микроконтроллеров по отладке через JTAG-порт;
- кабель, совместимый с кабелем Wiggler фирмы McGraigor Systems; этот кабель позволяет осуществлять внутрисхемное программирование подключенной к микроконтроллерам Flash-памяти с использованием возможностей JTAG-порта.

В итоге несложный подсчет показывает, что современному разработчику приходится иметь на своем столе до пяти различных кабелей для программирования устройств, что, согласитесь, доставляет некоторые неудобства. Именно поэтому автором и был разработан универсальный кабель, имитирующий работу каждого из выше перечисленных при определенной комбинации сигналов на входе (в самом простом, и, пожалуй, наиболее удобном случае, на эти входы может быть подключен переключатель или енкодер выдающий на выходе двоичный код текущего положения движка). Кроме того, огромным преимуществом рассматриваемого в статье кабеля перед всеми стандартными является то, что он построен на базе микросхемы программируемой логики, для программирования которой не требуется никаких других кабелей, что позволяет с легкостью добавлять новые типы кабелей по мере их появления (конечно, в разумных пределах, поскольку возникает естественное ограничение, исходя из объема применяемой микросхемы программируемой логики).

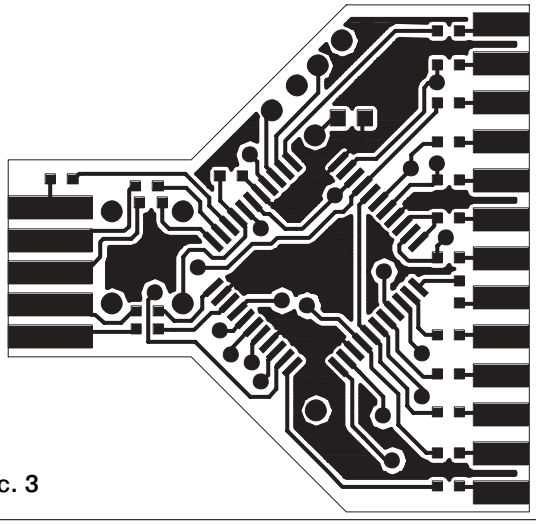

Рис. 3

Рассмотрим более подробно конструкцию кабеля, схема которого изображена на рис. 1. В основе конструкции лежит микросхема программируемой логики EPM3032ATC44-10 семейства MAX3000A фирмы Altera (DD1 на схеме). Эта микросхема содержит 32 логиче их ячейки, каждая из которых состоит из одного триггера и некоторого количества логических элементов. Питание микросхемы осуществляется от стабилизатора напряжения, собранного на линейном стабилизаторе SCI7810YDA со включенным последовательно его общему проводу диодом VD1. Выхо дное напряжения такого стабилизатора лежит в диапазоне 3,3–3,6 В, что соответствует номинальному рабочему напряжению микросхемы программируемой логики. Точки P1 и P2 на схеме соответствуют просто отверстиям в плате для подключения входного питающего напряжения. Само собой, схема питания может быть собрана любым образом, единственное требование к ней заключается в том, что она должна обеспечивать напряжение питания схемы около 3,3 В при токе до 50 мА. Здесь следует отметить, что при создании данной схемы автор ис-

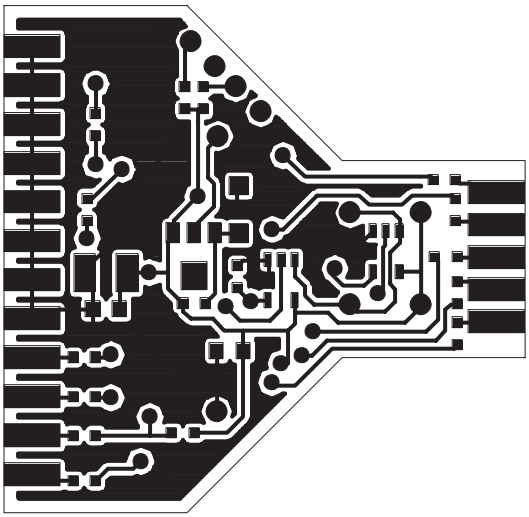

Рис. 4

пользовал исключительно то, что имелось под рукой, а также исходил из соображений минимизации размеров (авторский вариант кабеля размещен в корпусе переходника DB25-DB9, см. чертеж печатной платы на рис. 2 и 3), поэтому повторение ее в неизменном виде может быть довольно сложным в силу малого распространения использованных в ней элементов.

Микросхема программируемой логики подключается 17 выводами ко всем сигнальным линиям параллельного порта персонального компьютера, и еще 10 ее выводов подключены к внешнему разъему. Кроме того, два вывода микросхемы подключены к светодиодам, которые служат для индикации состояния программатора. Отдельно следует остановиться на назначении элементов DD2, DD3, SA1 и подключенных к ним компонентов. Поскольку микросхема программируемой логики требует начального программирования, а делать для этого дополнительный кабель и программировать им универсальный кабель — не совсем элегантное решение, в схему была включена цепь, позволяющая осуществлять программирование использованной в схеме кабеля микросхемы без каких бы то ни было внешних устройств. Для этого достаточно просто нажать кнопку SA1, и кабель будет распознан средой Max+Plus II фирмы Altera как ByteBlasterMV, но в качестве программируемого устройства будет выступать сама установленная в нем микросхема DD1. Осуществляется это за счет того, что при нажатии кнопки буферная микросхема DD2 подключает вход TCk микросхемы DD1 к соответствующему выводу параллельного порта компьютера, а микросхема DD3 подключает выход TDO микросхемы DD1 к выводу 11 LPT-порта, что соответствует подключению, используемому в кабеле Altera ByteBlasterMV. Кроме того, чтобы кабель был правильно распознан, требуется соответствие принятой схеме детектирования, что обеспечивают резисторы R32—R34. Микросхема DD1 должна либо быть чистой, либо прошивка в ней должна соответствующим образом реагировать на нажатие кнопки SB1. В качестве элементов DD2, DD3 можно использовать не только мало распространенные в России микросхемы TinyLogic, но и просто любой двух- или более разрядный буферный элемент, заменив им обе микросхемы.

Ниже приведен пример прошивки микросхемы для рассмотренных пяти типов кабелей:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity JTAGCable is
  Port(LPT01: in std_logic;
       LPT02: in std_logic;
       LPT03: in std_logic;
       LPT04: in std_logic;
       LPT05: in std_logic;
       LPT06: in std_logic;
       LPT07: in std_logic;
       LPT08: in std_logic;
       LPT09: in std_logic;
       LPT10: out std_logic;
       LPT11: out std_logic;
       LPT12: out std_logic;
       LPT13: out std_logic;
       LPT14: in std_logic;
       LPT15: out std_logic;
       LPT16: in std_logic;
       LPT17: in std_logic;
       CableType: in std_logic_vector (3 downto 0);
       JTAG: in std_logic;
       TMS: out std_logic;
       TCk: out std_logic;
       TClk: inout std_logic;
       TDI: out std_logic;
       TDO: inout std_logic;
       Rst: out std_logic;
       LEDA: out std_logic;
       LEDB: out std_logic
       );
end JTAGCable;

architecture behavioral of JTAGCable is

begin
  process(LPT01, LPT02, LPT03, LPT04, LPT05, LPT06,
LPT07, LPT08, LPT09, LPT14, LPT16, LPT17,
CableType, JTAG, TClk, TDO)
  begin
   if (JTAG='0') then
    case CableType(3 downto 0) is
     when «0000» =>
      TMS<= LPT03;
      TCk<= LPT04;
      LEDA<= LPT04;
      TClk<= LPT06;
      TDI<= LPT05;
      LEDB<= LPT05;
      TDO<= 'Z';
      Rst<= not LPT02;
      LPT10<= '1';
      LPT11<= TDO;
      LPT12<= '1';
      LPT13<= '1';
      LPT15<= '1';
     when «0001» =>
      LEDA<= LPT14;
      if (LPT14='0') then
       TMS<= LPT03;
       TCk<= LPT02;
       TDI<= LPT08;
       LEDB<= LPT08;
      else
       TMS<= '1';
       TCk<= '1';
       TDI<= '1';
       LEDB<= '1';
      end if;
      TClk<= 'Z';
      TDO<= 'Z';
      Rst<= '1';
      LPT10<= LPT07;
      LPT11<= TDO;
      LPT12<= LPT09;
      LPT13<= TClk;
      LPT15<= '1';
     when «0010» =>
      LEDA<= LPT05;
      if (LPT05='0') then
       TMS<= LPT04;
       TCk<= LPT03;
       TDI<= LPT02;
       LEDB<= LPT02;
      else
       TMS<= '1';
       TCk<= '1';
       TDI<= '1';
       LEDB<= '1';
      end if;
      TClk<= 'Z';
      if (LPT06='0') then
       TDO<= '0';
       LPT13<= '0';
```

```
  else
    TDO<= 'Z';
    LPT13<= TDO;
  end if;
  Rst<= '1';
  LPT10<= '1';
  LPT11<= LPT08;
  LPT12<= LPT08;
  LPT15<= '1';
 when «0011» =>
  TDO<= 'Z';
  LEDA<= LPT17;
  if (LPT17='0') then
    TMS<= LPT03;
    TCk<= LPT04;
    TDI<= LPT02;
    LEDB<= LPT02;
    LPT12<= TDO;
  else
    TMS<= '1';
    TCk<= '1';
    TDI<= '1';
    LEDB<= '1';
    LPT12<= '1';
  end if;
  if (LPT14='0') then
    TClk<= LPT05;
    Rst<= LPT01;
  else
    TClk<= 'Z';
    Rst<= '1';
  end if;
  LPT10<= '1';
  LPT11<= '1';
  LPT13<= '1';
  LPT15<= '1';
 when «0100» =>
  TDO<= 'Z';
  if (LPT04='0') then
    TMS<= LPT09;
    LPT10<= TDO;
  else
    TMS<= '1';
    LPT10<= '1';
  end if;
  LEDA<= LPT05;
  if (LPT05='0') then
    TCk<= LPT06;
    TDI<= LPT07;
    LEDB<= LPT07;
  else
    TCk<= '1';
    TDI<= '1';
    LEDB<= '1';
  end if;
  TClk<= 'Z';
  Rst<= '1';
  LPT11<= LPT03;
  LPT12<= LPT02;
  LPT13<= '1';
  LPT15<= '1';
 when others =>
  LPT10<= '1';
  LPT11<= '1';
  LPT12<= '1';
  LPT13<= '1';
  LPT15<= '1';
  TMS<= '1';
  TCk<= '1';
  TClk<= 'Z';
  TDI<= '1';
  TDO<= 'Z';
  Rst<= '1';
  LEDA<= '1';
  LEDB<= '1';
 end case;
else
 LPT10<= LPT07;
  LPT11<= 'Z';
  LPT12<= LPt09;
  LPT13<= '1';
  LPT15<= '1';
  TMS<= '1';
  TCk<= '1';
  TClk<= 'Z';
  TDI<= '1';
  TDO<= 'Z';
  Rst<= '1';
  LEDA<= '1';
  LEDB<= '1';
 end if;
end process;
end behavioral;
```

Данная прошивка написана на языке VHDL и рассчитана на автоматическое определение типа кабеля путем анализа входных сигналов CableType. Прошивка компилируется в среде Max+Plus II фирмы Altera и программируется указанным выше способом в микросхему, установленную в кабеле. Следует только не забыть правильно расставить сигналы по выводам микросхемы. Логику функционирования прошивки легко понять даже человеку, абсолютно не знакомому с языком VHDL, так что добавление новых типов кабелей не составит никакого труда и для тех, кто никогда не работал с программируемой логикой.

Алексей Сигаев,
sigaev@geolink.ru