One Dollar Dongle

One Dollar Dongle

Published 2003

Table of Contents

List of Figures

List of Examples

Chapter 1. One Dollar Dongle

1.1. ODD Version 1.0

One-Layer PCB, minimal components. Supply 3.3V from target connector. LPT connector is header 2x13 (use standard LPT extension cable).

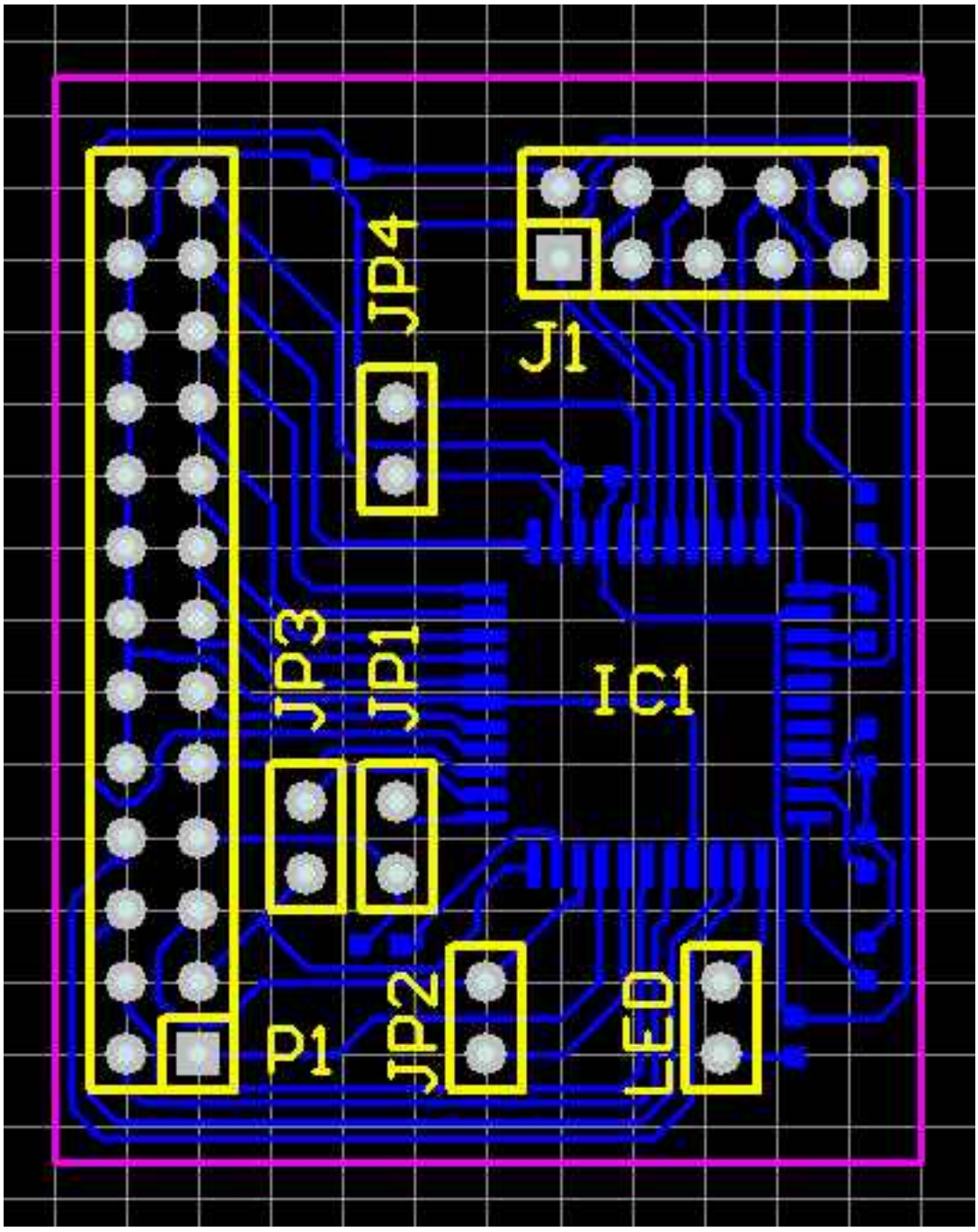

Figure 1.1. ODD ver 1.0 PCB

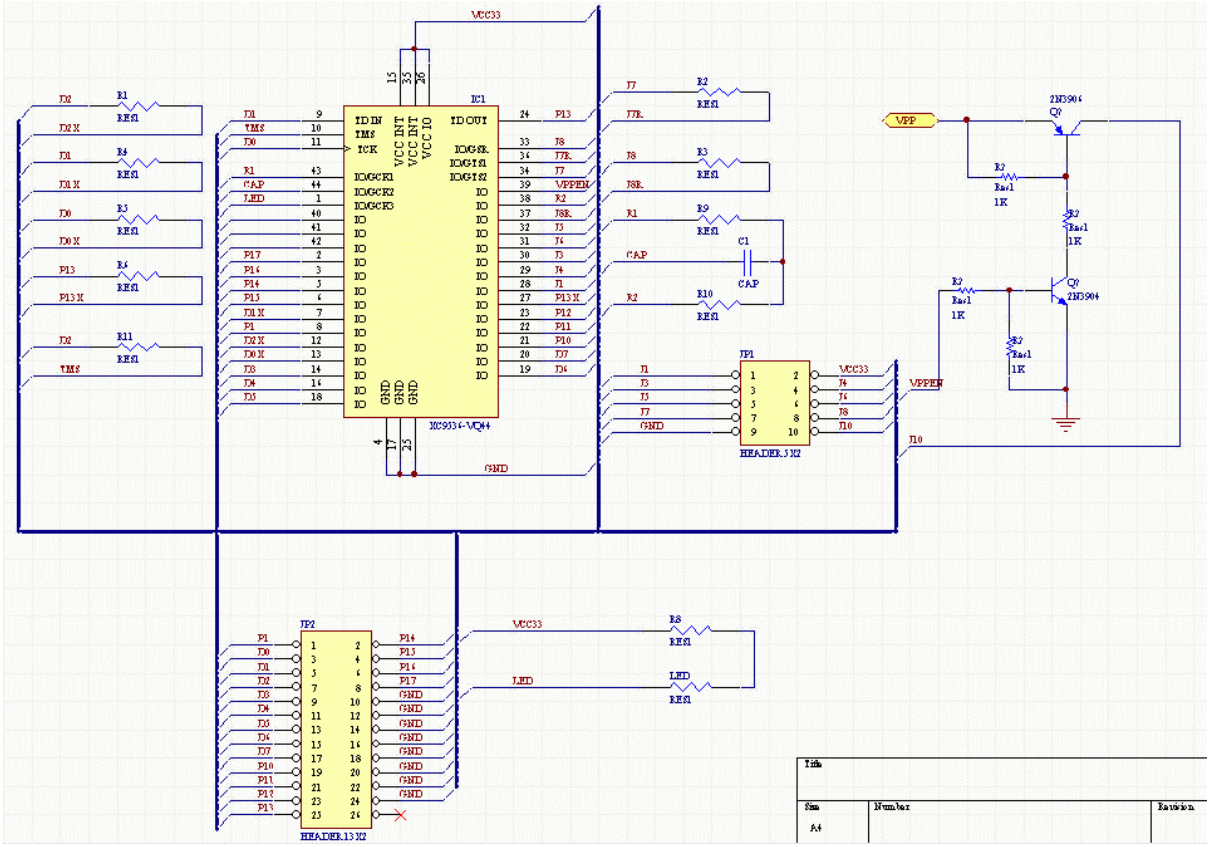

Figure 1.2. ODD Schematics

1.2. ODD Configurations

1.2.1. CheapI2C

http://warmcat.com/milksop/cheapi2c.html Original I2C interface for I2C monitoring (free Linux software). Design has capabilities to be used as I2C master interface.

Figure 1.3. CheapI2C

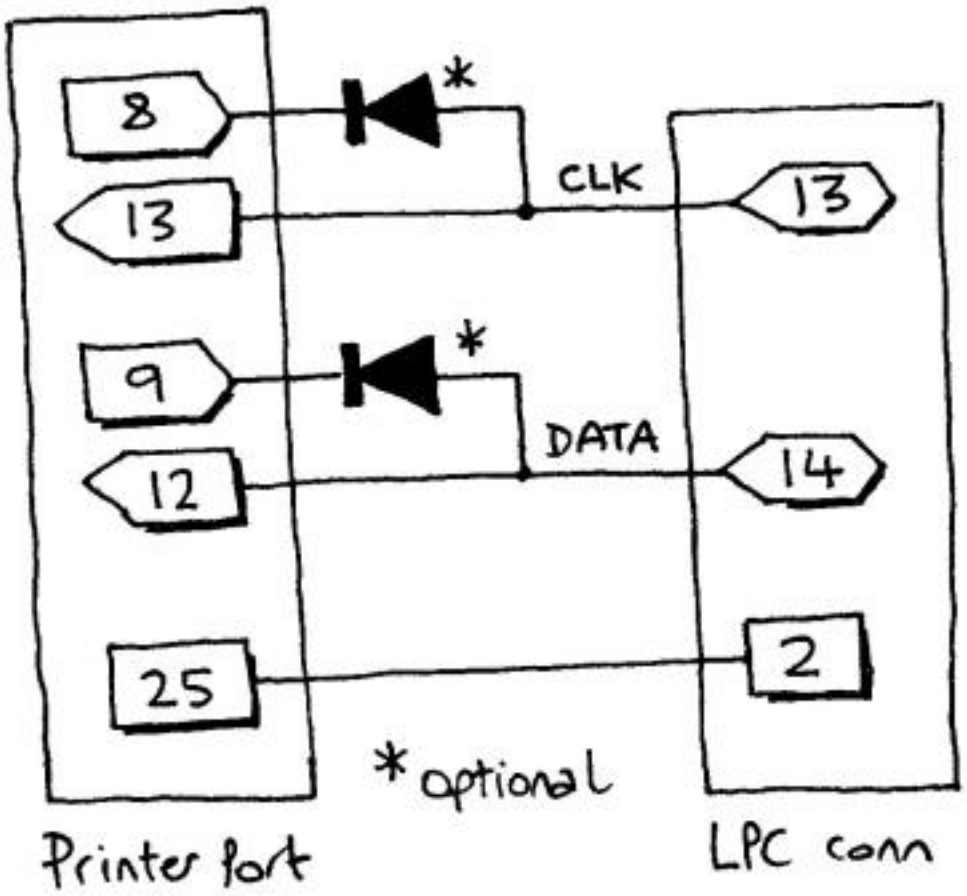

Figure 1.4. CheapI2C Schematics

Example 1.1. cheapi2c.vhd

```
-- Copyright 2003 Nugis Foundation.
-- Based on documentation from www.lancos.com
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity cheapi2c is Port (
  SDA_INOUT_J14 : inout std_logic;
  SCL_INOUT_J13 : inout std_logic;
  SDA_IN_D7 : in std_logic;
  SCL_IN_D6 : in std_logic;
  SDA_OUT_P12 : out std_logic;
  SCL_OUT_P13 : out std_logic
  );
end cheapi2c;

architecture Behavioral of cheapi2c is
begin
  SDA_INOUT_J14 <= '0' when (SDA_IN_D7 = '0') else 'Z';
  SCL_INOUT_J13 <= '0' when (SCL_IN_D6 = '0') else 'Z';
  SDA_OUT_P12 <= SDA_INOUT_J14;
  SCL_OUT_P13 <= SCL_INOUT_J13;
end Behavioral;
```

Example 1.2. odd_cheapi2c.vhd

```
-- Copyright 2003 Nugis Foundation
library IEEE;

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

use work.ALL;

entity odd_cheapi2c is Port (
  -- LPT Port Data (Pin 2..9)
  D0 : in std_logic;
  D1 : in std_logic;
  D2 : in std_logic;
  D3 : in std_logic;
  D4 : in std_logic;
  D5 : in std_logic;
  D6 : in std_logic;
  D7 : in std_logic;
  -- LPT Port Control/Status ( Pn = Pin(n) )
  P10 : in std_logic;
  P11 : in std_logic;
  P12 : out std_logic;
  P13 : inout std_logic;
  P14 : in std_logic;
  P15 : in std_logic;
  P16 : in std_logic;
  P17 : in std_logic;
  -- Programming Connector 10-Way IDC
  J1  : in std_logic;
  J3  : in std_logic;
  J4  : in std_logic;
  J5  : in std_logic;
  J6  : in std_logic;
  J7  : inout std_logic;
  J7R : in std_logic; -- Pull-up for J7
  J8  : inout std_logic;
  J8R : in std_logic; -- Pull-up for J8
  -- R/C Oscillator
  R1  : in std_logic;
  R2  : in std_logic;
  CAP : in std_logic;
  -- Reserved Pins?
  res1: in std_logic;

  -- LED (low is LIT)
  LED : out std_logic;
  -- "Dummy" (dont care) output
  DUMMY : out std_logic
  );
end odd_cheapi2c;

architecture Behavioral of odd_cheapi2c is

COMPONENT cheapi2c
        PORT(
                SDA_IN_D7 : IN std_logic;
                SCL_IN_D6 : IN std_logic;
                SDA_INOUT_J14 : INOUT std_logic;
                SCL_INOUT_J13 : INOUT std_logic;
                SDA_OUT_P12 : OUT std_logic;
                SCL_OUT_P13 : OUT std_logic
                );
        END COMPONENT;

  COMPONENT odd_dummy PORT(
    TDI_IN : IN std_logic;
    TDO_IN : IN std_logic;
    TCK_IN : IN std_logic;
    TMS_IN : IN std_logic;
    JTAG_DUMMY : OUT std_logic);
    END COMPONENT;
```

```
begin

        Inst_cheapi2c: cheapi2c PORT MAP(
                SDA_INOUT_J14 => J8,
                SCL_INOUT_J13 => J7,
                SDA_IN_D7 => D7,
                SCL_IN_D6 => D6,
                SDA_OUT_P12 => P12,
                SCL_OUT_P13 => P13);


  -- Dummy instance to keep JTAG pins as input
  Inst_odd_dummy: ODD_dummy PORT MAP(
    TDI_IN => D0,
    TDO_IN => D1,
    TCK_IN => D2,
    TMS_IN => P13,
    JTAG_DUMMY => DUMMY);

  -- Make LED Lit always ?
  LED <= '0';

end Behavioral;
```

1.2.2. EasyI2C

Software: PonyProg

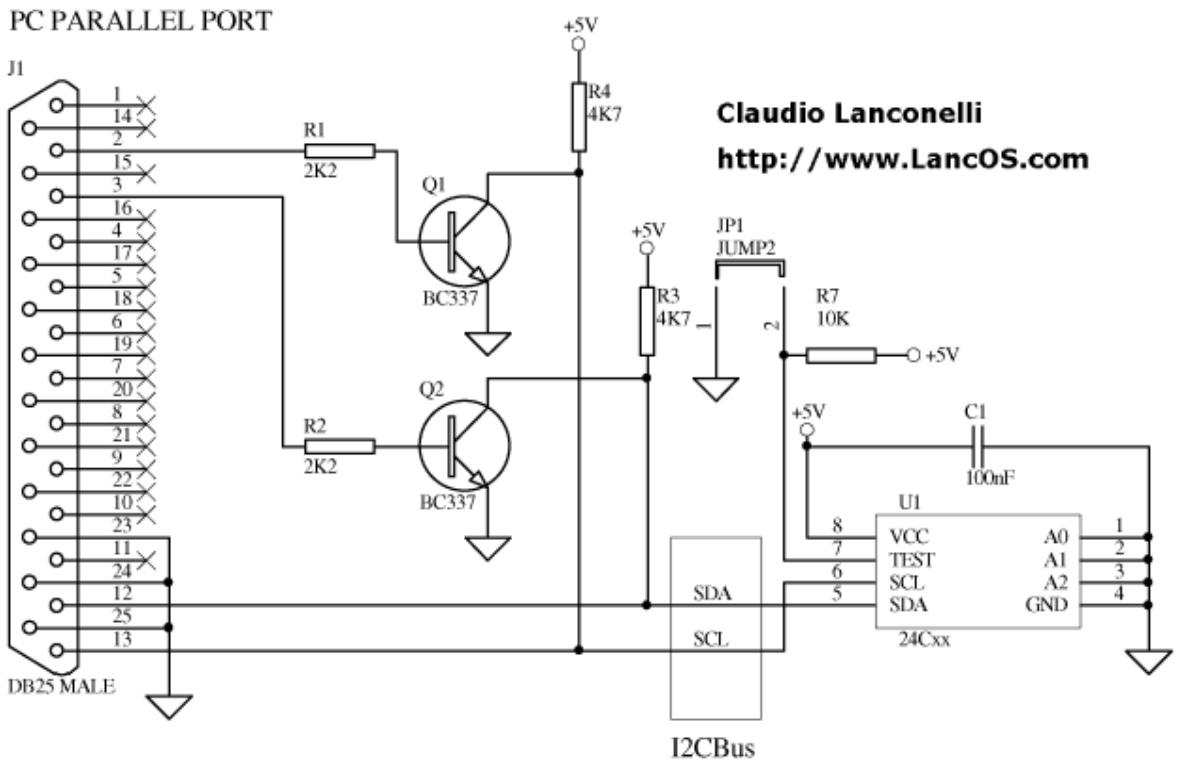

Figure 1.5. EasyI2C Schematics

Example 1.3. easyi2c.vhd

```
-- Copyright 2003 Nugis Foundation.
-- Based documentation from www.lancos.com
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity easyi2c is Port (
  SDA_INOUT : inout std_logic;
  SCL_INOUT : inout std_logic;
  SDA_IN_D1 : in std_logic;
  SCL_IN_D0 : in std_logic;
  SDA_OUT_P12 : out std_logic;
  SCL_OUT_P13 : out std_logic
  );
end easyi2c;

architecture Behavioral of easyi2c is
begin
  SDA_INOUT <= '0' when (SDA_IN_D1 = '0') else 'Z';
  SCL_INOUT <= '0' when (SCL_IN_D0 = '0') else 'Z';
  SDA_OUT_P12 <= SDA_INOUT;
  SCL_OUT_P13 <= SCL_INOUT;
end Behavioral;
```

Example 1.4. odd_easyi2c.vhd

```
-- Copyright 2003 Nugis Foundation
library IEEE;

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

use work.ALL;

entity odd_easyi2c is Port (
  -- LPT Port Data (Pin 2..9)
  D0 : in std_logic;
  D1 : in std_logic;
  D2 : in std_logic;
  D3 : in std_logic;
  D4 : in std_logic;
  D5 : in std_logic;
  D6 : in std_logic;
  D7 : in std_logic;
  -- LPT Port Control/Status ( Pn = Pin(n) )
  P10 : in std_logic;
  P11 : in std_logic;
  P12 : out std_logic;
  P13 : inout std_logic;
  P14 : in std_logic;
  P15 : in std_logic;
  P16 : in std_logic;
  P17 : in std_logic;

  -- Programming Connector 10-Way IDC
  J1  : in std_logic;
  J3  : in std_logic;
  J4  : in std_logic;
  J5  : in std_logic;
  J6  : in std_logic;
  J7  : inout std_logic;
  J7R : out std_logic; -- Pull-up for J7
  J8  : inout std_logic;
  J8R : out std_logic; -- Pull-up for J8
  -- R/C Oscillator
  R1  : in std_logic;
  R2  : in std_logic;
  CAP : in std_logic;
  -- Reserved Pins?
  res1: in std_logic;
```

```
  -- LED (low is LIT)
  LED : out std_logic;
  -- "Dummy" (dont care) output
  DUMMY : out std_logic
  );
end odd_easyi2c;

architecture Behavioral of odd_easyi2c is

  COMPONENT easyi2c PORT(
    SDA_IN_D1 : IN std_logic;
    SCL_IN_D0 : IN std_logic;
    SDA_INOUT : INOUT std_logic;
    SCL_INOUT : INOUT std_logic;
    SDA_OUT_P12 : OUT std_logic;
    SCL_OUT_P13 : OUT std_logic
  );
  END COMPONENT;

  COMPONENT odd_dummy PORT(
    TDI_IN : IN std_logic;
    TDO_IN : IN std_logic;
    TCK_IN : IN std_logic;
    TMS_IN : IN std_logic;
    JTAG_DUMMY : OUT std_logic
  );
  END COMPONENT;

begin
  -- easy I2C
  Inst_easyi2c: easyi2c PORT MAP(
    SDA_INOUT => J8,
    SCL_INOUT => J7,
    SDA_IN_D1 => D1,
    SCL_IN_D0 => D0,
    SDA_OUT_P12 => P12,
    SCL_OUT_P13 => P13
  );
  -- enable pull-ups on J7 and J8
  J7R <= '1';
  J8R <= '1';

  -- Dummy instance to keep JTAG pins as input
  Inst_odd_dummy: ODD_dummy PORT MAP(
    TDI_IN => D0,
    TDO_IN => D1,
    TCK_IN => D2,
    TMS_IN => P13,
    JTAG_DUMMY => DUMMY
  );

  -- Make LED Lit always ?
  LED <= '0';

end Behavioral;
```

1.2.3. CheapLPC

LPC Flash ROM Programmer. GPL Software.

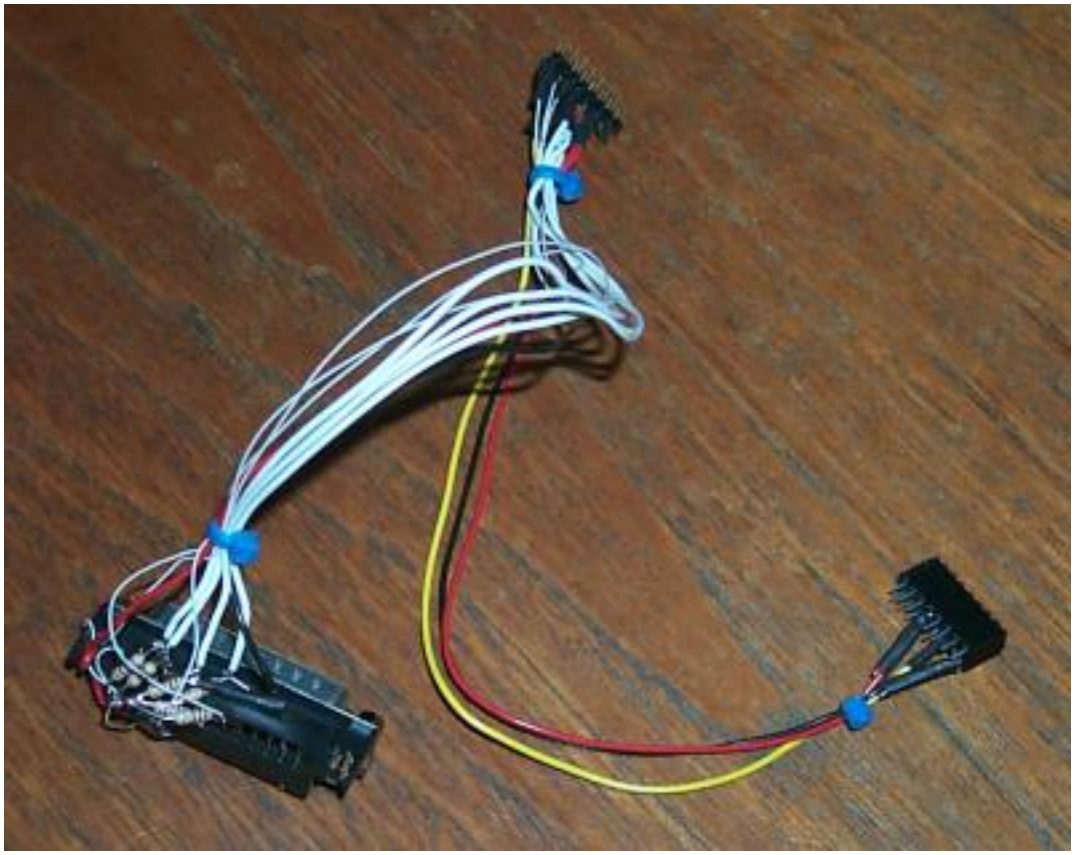

Figure 1.6. CheapLPC

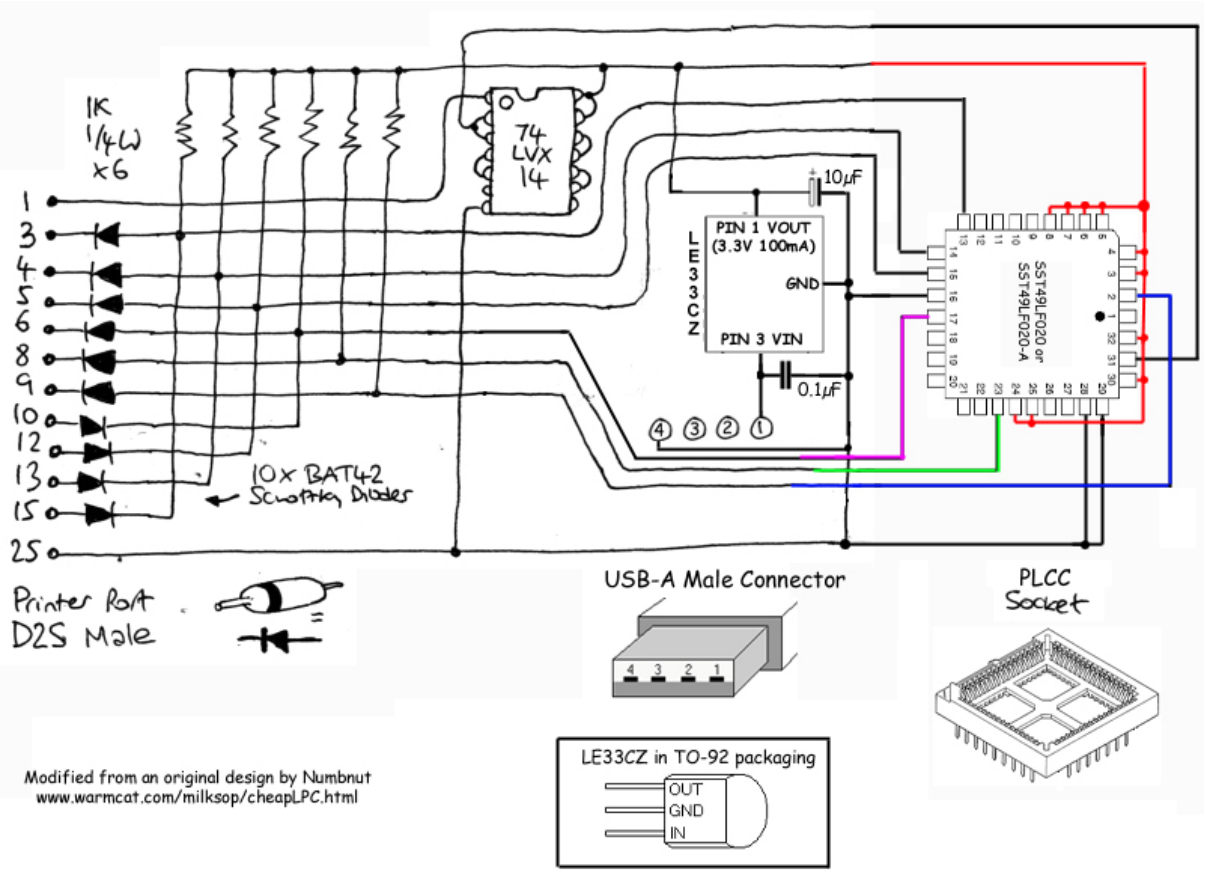

Figure 1.7. CheapLPC Schematics

1.2.4. SMX3200

http://www.summitmicro.com Note: The 10V step-up is not available, need external supply (if required for programming).

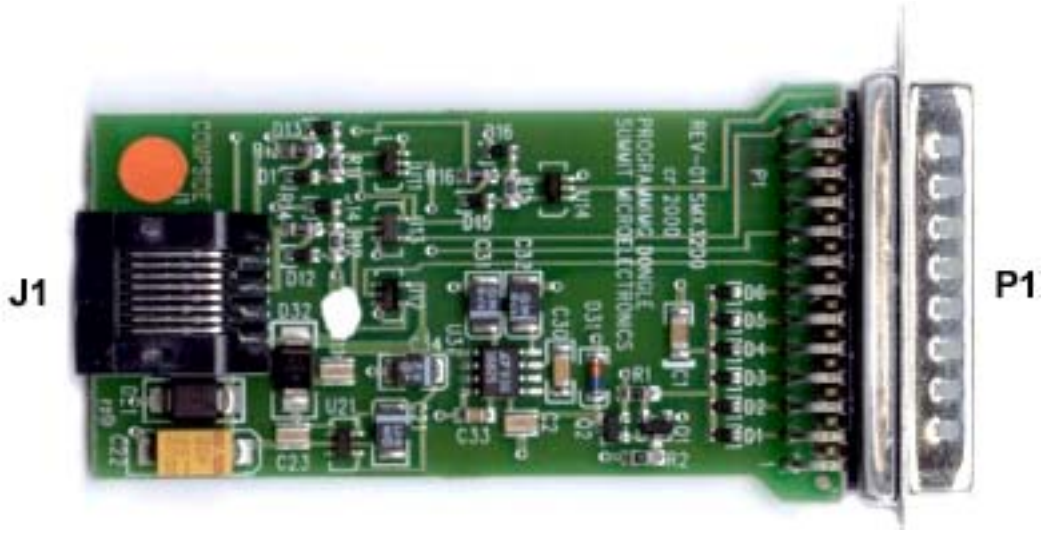

Figure 1.8. SMX3200 Board

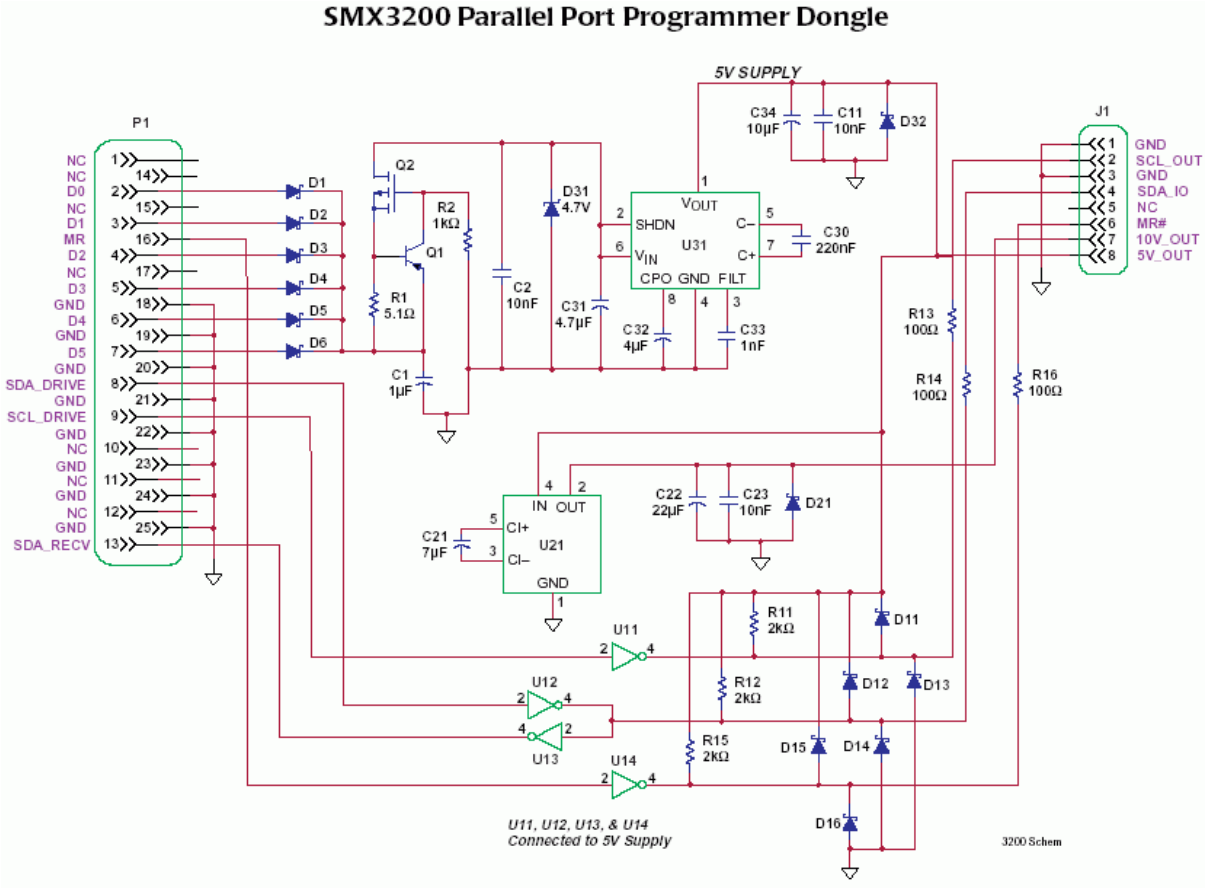

Figure 1.9. SMX3200 Schematics

Example 1.5. smx3200.vhd

```
-- Copyright 2003 Nugis Foundation.
-- Based on documentation from www.summitmicro.com
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity smx3200 is Port (
  SDA_INOUT_J4 : inout std_logic;
  SCL_OUT_J2 : out std_logic;
  MR_OUT_J6 : out std_logic;
  SDA_IN_D6 : in std_logic;
  SDA_OUT_P13 : out std_logic;
  SCL_IN_D7 : in std_logic;
  MR_IN_P16: in std_logic
  );
end smx3200;

architecture Behavioral of smx3200 is
begin
  -- all inverted connections
  SDA_INOUT_J4 <= '0' when (SDA_IN_D6 = '1') else 'Z';
  SDA_OUT_P13 <= not SDA_INOUT_J4;
  SCL_OUT_J2 <= not SCL_IN_D7;
  MR_OUT_J6 <= not MR_IN_P16;
end Behavioral;
```

Example 1.6. odd_smx3200.vhd

```
-- Copyright 2003 Nugis Foundation
library IEEE;

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

use work.ALL;

entity odd_SMX3200 is Port (
  -- LPT Port Data (Pin 2..9)
  D0 : in std_logic;
  D1 : in std_logic;
  D2 : in std_logic;
  D3 : in std_logic;
  D4 : in std_logic;
  D5 : in std_logic;
  D6 : in std_logic;
  D7 : in std_logic;
  -- LPT Port Control/Status ( Pn = Pin(n) )
  P10 : in std_logic;
  P11 : in std_logic;
  P12 : in std_logic;
  P13 : out std_logic;
  P14 : in std_logic;
  P15 : in std_logic;
  P16 : in std_logic;
  P17 : in std_logic;
  -- Programming Connector 10-Way IDC
  J1  : in std_logic;
  J3  : in std_logic;
  J4  : in std_logic;
  J5  : in std_logic;
  J6  : out std_logic;
  J7  : out std_logic;
  J7R : in std_logic; -- Pull-up for J7
  J8  : inout std_logic;
  J8R : out std_logic; -- Pull-up for J8
  -- R/C Oscillator
  R1  : in std_logic;
  R2  : in std_logic;
  CAP : in std_logic;
  -- Reserved Pins?
  res1: in std_logic;

  -- LED (low is LIT)
  LED : out std_logic;
  -- "Dummy" (dont care) output
  DUMMY : out std_logic
  );
end odd_SMX3200;

architecture Behavioral of odd_SMX3200 is

  COMPONENT smx3200 PORT(
    SDA_IN_D6 : IN std_logic;
    SDA_OUT_P13 : IN std_logic;
    SCL_IN_D7 : IN std_logic;
    MR_IN_P16 : IN std_logic;
    SDA_INOUT_J4 : INOUT std_logic;
    SCL_OUT_J2 : OUT std_logic;
    MR_OUT_J6 : OUT std_logic);
  END COMPONENT;

  COMPONENT odd_dummy PORT(
    TDI_IN : IN std_logic;
    TDO_IN : IN std_logic;
    TCK_IN : IN std_logic;
    TMS_IN : IN std_logic;
    JTAG_DUMMY : OUT std_logic);
  END COMPONENT;
```

```
begin
  --
  Inst_smx3200: smx3200 PORT MAP(
    SDA_INOUT_J4 => J8,
    SCL_OUT_J2 => J7,
    MR_OUT_J6 => J6,
    SDA_IN_D6 => D6,
    SDA_OUT_P13 => P13,
    SCL_IN_D7 => D7,
    MR_IN_P16 => P16
  );
  -- Enable Pull-ups
  J8R <= '1';

  -- Dummy instance to keep JTAG pins as input
  Inst_odd_dummy: ODD_dummy PORT MAP(
    TDI_IN => D0,
    TDO_IN => D1,
    TCK_IN => D2,
    TMS_IN => P13,
    JTAG_DUMMY => DUMMY
  );

  -- Make LED Lit always ?
  LED <= '0';

end Behavioral;
```

1.2.5. Philips LPT I2C Interface

Original I2C Interface from Philips BBS.

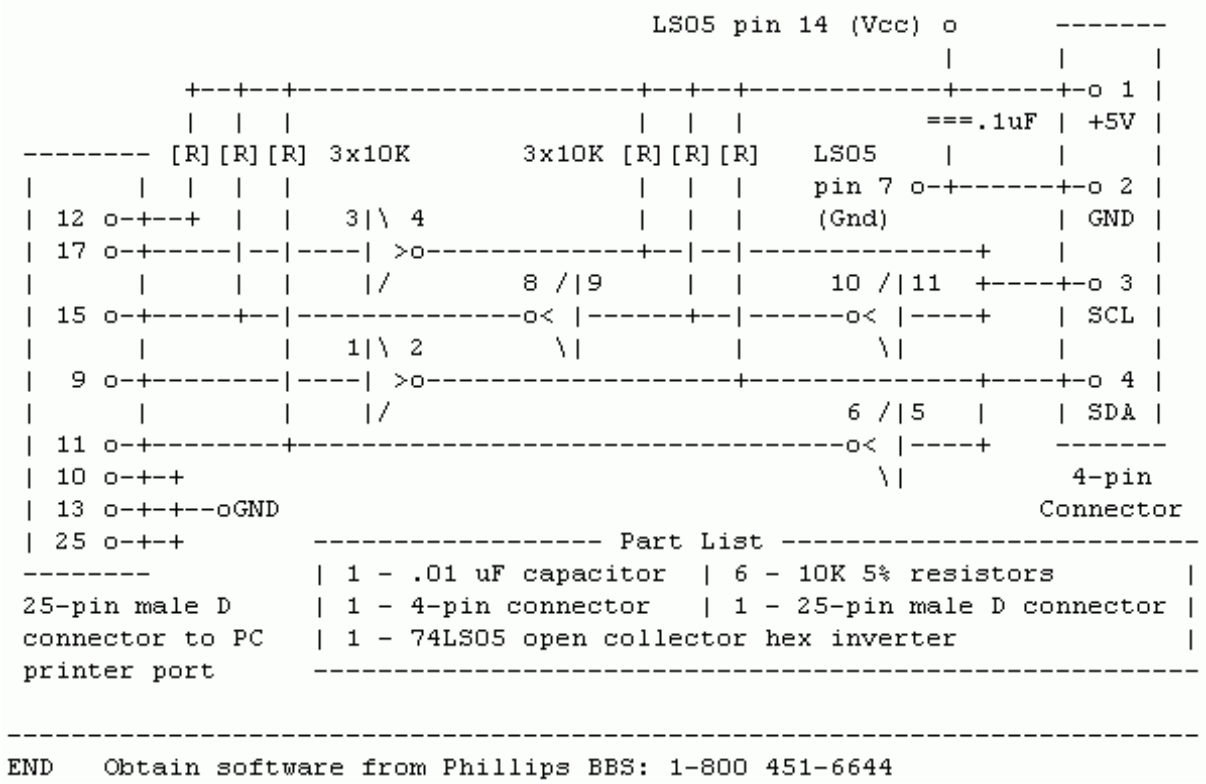

Figure 1.10. Philips I2C LPT Interface Schematics

Example 1.7. i2clpt.vhd

```
-- Copyright 2003 Nugis Foundation.
-- LPT I2C Bus Interface (original from Philips)
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity i2clpt is Port (
  SDA_INOUT_J4 : inout std_logic;
  SCL_INOUT_J3 : inout std_logic;
  SDA_IN_D7 : in std_logic;
  SCL_IN_P17 : in std_logic;
  SDA_OUT_P11 : out std_logic;
  SCL_OUT_P15 : out std_logic
  );
end i2clpt;

architecture Behavioral of i2clpt is
begin
  SDA_INOUT_J4 <= '0' when (SDA_IN_D7 = '0') else 'Z';
  SCL_INOUT_J3 <= '0' when (SCL_IN_P17 = '0') else 'Z';
  SDA_OUT_P11 <= SDA_INOUT_J4;
  SCL_OUT_P15 <= SCL_INOUT_J3;
end Behavioral;
```

Example 1.8. odd_i2clpt.vhd

```
-- Copyright 2003 Nugis Foundation
library IEEE;

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

use work.ALL;

entity odd_i2clpt is Port (
  -- LPT Port Data (Pin 2..9)
  D0 : in std_logic;
  D1 : in std_logic;
  D2 : in std_logic;
  D3 : in std_logic;
  D4 : in std_logic;
  D5 : in std_logic;
  D6 : in std_logic;
  D7 : in std_logic;
  -- LPT Port Control/Status ( Pn = Pin(n) )
  P10 : in std_logic;
  P11 : out std_logic;
  P12 : in std_logic;
  P13 : in std_logic;
  P14 : in std_logic;
  P15 : out std_logic;
  P16 : in std_logic;
  P17 : in std_logic;
  -- Programming Connector 10-Way IDC
  J1  : in std_logic;
  J3  : in std_logic;
  J4  : in std_logic;
  J5  : in std_logic;
  J6  : in std_logic;
  J7  : inout std_logic;
  J7R : out std_logic; -- Pull-up for J7
  J8  : inout std_logic;
  J8R : out std_logic; -- Pull-up for J8
  -- R/C Oscillator
  R1  : in std_logic;
```

```
  R2  : in std_logic;
  CAP : in std_logic;
  -- Reserved Pins?
  res1: in std_logic;

  -- LED (low is LIT)
  LED : out std_logic;
  -- "Dummy" (dont care) output
  DUMMY : out std_logic
  );
end odd_i2clpt;

architecture Behavioral of odd_i2clpt is

  COMPONENT i2clpt PORT(
    SDA_IN_D7 : IN std_logic;
    SCL_IN_P17 : IN std_logic;
    SDA_INOUT_J4 : INOUT std_logic;
    SCL_INOUT_J3 : INOUT std_logic;
    SDA_OUT_P11 : OUT std_logic;
    SCL_OUT_P15 : OUT std_logic);
  END COMPONENT;

  COMPONENT odd_dummy PORT(
    TDI_IN : IN std_logic;
    TDO_IN : IN std_logic;
    TCK_IN : IN std_logic;
    TMS_IN : IN std_logic;
    JTAG_DUMMY : OUT std_logic);
    END COMPONENT;

begin
  Inst_i2clpt: i2clpt PORT MAP(
    SDA_INOUT_J4 => J8,
    SCL_INOUT_J3 => J7,
    SDA_IN_D7 => D7,
    SCL_IN_P17 => P17,
    SDA_OUT_P11 => P11,
    SCL_OUT_P15 => P15);

  -- Enable pull-ups
  J8R <= '1';
  J7R <= '1';

  -- Dummy instance to keep JTAG pins as input
  Inst_odd_dummy: ODD_dummy PORT MAP(
    TDI_IN => D0,
    TDO_IN => D1,
    TCK_IN => D2,
    TMS_IN => P13,
    JTAG_DUMMY => DUMMY);

  -- Make LED Lit always ?
  LED <= '0';

end Behavioral;
```

1.2.6. MacGraigor JTAG Wiggler

Example 1.9. wiggler.vhd

```
-- Copyright 2003 Nugis Foundation.
-- MacGraigor JTAG Wiggler
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity wiggler is Port (
```

```
  TCK_OUT_J9   : out std_logic;
  TDI_OUT_J5   : out std_logic;
  TDO_IN_J13   : in std_logic;
  TMS_OUT_J7   : out std_logic;
  TRST_OUT_J15 : out std_logic;

  TCK_IN_D2   : in std_logic;
  TDI_IN_D3   : in std_logic;
  TDO_OUT_P11 : out std_logic;
  TMS_IN_D1   : in std_logic;
  TRST_IN_D0  : in std_logic);
end wiggler;

architecture Behavioral of wiggler is
begin
  -- HC244 is non-inverting buffer
  TCK_OUT_J9 <= TCK_IN_D2;
  TDI_OUT_J5 <= TDI_IN_D3;
  TMS_OUT_J7 <= TMS_IN_D1;
  TDO_OUT_P11 <= TDO_IN_J13;
  TRST_OUT_J15 <= not TRST_IN_D0;
end Behavioral;
```

Example 1.10. odd_wiggler.vhd

```
-- Copyright 2003 Nugis Foundation
library IEEE;

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

use work.ALL;

entity odd_wiggler is Port (
  -- LPT Port Data (Pin 2..9)
  D0 : in std_logic;
  D1 : in std_logic;
  D2 : in std_logic;
  D3 : in std_logic;
  D4 : in std_logic;
  D5 : in std_logic;
  D6 : in std_logic;
  D7 : in std_logic;
  -- LPT Port Control/Status ( Pn = Pin(n) )
  P10 : in std_logic;
  P11 : out std_logic;
  P12 : in std_logic;
  P13 : in std_logic;
  P14 : in std_logic;
  P15 : in std_logic;
  P16 : in std_logic;
  P17 : in std_logic;
  -- Programming Connector 10-Way IDC
  J1  : out std_logic;
  J3  : out std_logic;
  J4  : out std_logic;
  J5  : in std_logic;
  J6  : in std_logic;
  J7  : out std_logic;
  J7R : in std_logic; -- Pull-up for J7
  J8  : in std_logic;
  J8R : in std_logic; -- Pull-up for J8
  -- R/C Oscillator
  R1  : in std_logic;
  R2  : in std_logic;
  CAP : in std_logic;
  -- Reserved Pins?
  res1: in std_logic;

  -- LED (low is LIT)
  LED : out std_logic;
```

```
  -- "Dummy" (dont care) output
  DUMMY : out std_logic
  );
end odd_wiggler;

architecture Behavioral of odd_wiggler is

  COMPONENT wiggler PORT(
    TDO_IN_J13 : IN std_logic;
    TDO_OUT_P11 : OUT std_logic;
    TCK_OUT_J9 : OUT std_logic;
    TDI_OUT_J5 : OUT std_logic;
    TMS_OUT_J7 : OUT std_logic;
    TRST_OUT_J15 : OUT std_logic;
    TCK_IN_D2 : IN std_logic;
    TDI_IN_D3 : IN std_logic;
    TMS_IN_D1 : IN std_logic;
    TRST_IN_D0 : IN std_logic);
  END COMPONENT;


  COMPONENT odd_dummy PORT(
    TDI_IN : IN std_logic;
    TDO_IN : IN std_logic;
    TCK_IN : IN std_logic;
    TMS_IN : IN std_logic;
    JTAG_DUMMY : OUT std_logic);
    END COMPONENT;

begin
  --
  Inst_wiggler: wiggler PORT MAP(
    TCK_OUT_J9 => J1,
    TDI_OUT_J5 => J7,
    TDO_IN_J13 => J8,
    TMS_OUT_J7 => J3,
    TRST_OUT_J15 => J4,
    TCK_IN_D2 => D2,
    TDI_IN_D3 => D3,
    TDO_OUT_P11 => P11,
    TMS_IN_D1 => D1,
    TRST_IN_D0 => D0);

  -- Dummy instance to keep JTAG pins as input
  Inst_odd_dummy: ODD_dummy PORT MAP(
    TDI_IN => D0,
    TDO_IN => D1,
    TCK_IN => D2,
    TMS_IN => P13,
    JTAG_DUMMY => DUMMY);

  -- Make LED Lit always ?
  LED <= '0';

end Behavioral;
```

1.2.7. Philips OM4777 I2C Adapter

This is an old bit-bang I2C interface, introduced by Philips. Several 'clones' do exist. Same configuration works for 'cloned' designs as well.

1.2.8. ST/Waferscale Flashlink

Programming adapter for Waferscale (later ST) programmable devices. This was the first at all configuration tested - only test for Programming adapter presence performed (Nugis Foundation does not yet have any devices to test with).

1.2.9. Low Voltage Trival PIC Programmer

This was the first configuration actually tested with first prototype of ODD. A PIC16F877 on flip-flop Board from EDTP was programed using David Tait's FPP Windows based PIC Programming software. Configuration was done with WebPack using Schematics Entry.

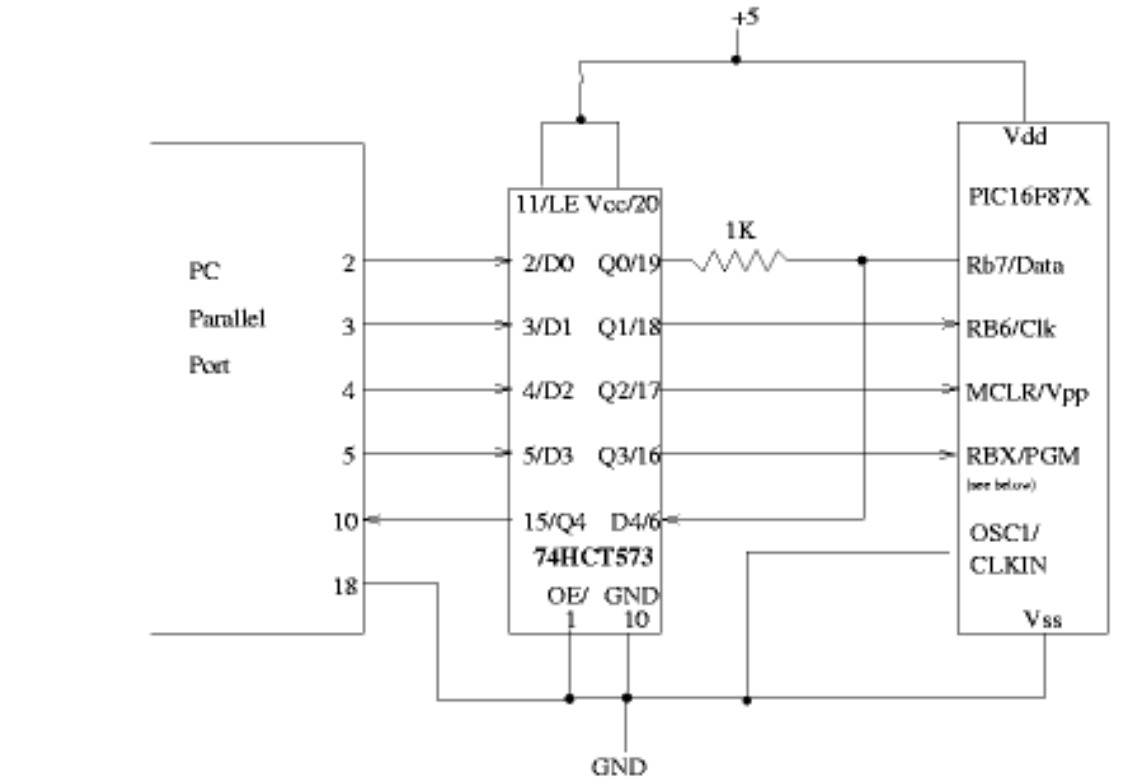

Figure 1.11. LV Trivial PIC Programmer

Example 1.11. lvtpp.vhd

```
-- Copyright 2003 Nugis Foundation.
-- Low Voltage Trival PIC Programmer
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity lvtpp is Port (
  DATA_INOUT   : inout std_logic;
  DATA_IN_D1   : in std_logic;
  DATA_OUT_P10 : out std_logic;
  CLK_OUT      : out std_logic;
  CLK_IN_D1    : in std_logic;
  MCLR_OUT     : out std_logic;
  MCLR_IN_D2   : in std_logic;
  PGM_OUT      : out std_logic;
  PGM_IN_D3    : in std_logic
```

```
  );
end lvtpp;

architecture Behavioral of lvtpp is
begin
  DATA_INOUT <= '0' when (DATA_IN_D1 = '0') else 'Z';
  DATA_OUT_P10 <= DATA_INOUT;
  CLK_OUT <= CLK_IN_D1;
  MCLR_OUT <= MCLR_IN_D2;
  PGM_OUT <= PGM_IN_D3;
end Behavioral;
```

Example 1.12. odd_lvtpp.vhd

```
-- Copyright 2003 Nugis Foundation
library IEEE;

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

use work.ALL;

entity odd_lvtpp is Port (
  -- LPT Port Data (Pin 2..9)
  D0 : in std_logic;
  D1 : in std_logic;
  D2 : in std_logic;
  D3 : in std_logic;
  D4 : in std_logic;
  D5 : in std_logic;
  D6 : in std_logic;
  D7 : in std_logic;
  -- LPT Port Control/Status ( Pn = Pin(n) )
  P10 : out std_logic;
  P11 : in std_logic;
  P12 : in std_logic;
  P13 : in std_logic;
  P14 : in std_logic;
  P15 : in std_logic;
  P16 : in std_logic;
  P17 : in std_logic;
  -- Programming Connector 10-Way IDC
  J1  : out std_logic;
  J3  : out std_logic;
  J4  : out std_logic;
  J5  : in std_logic;
  J6  : in std_logic;
  J7  : in std_logic;
  J7R : in std_logic; -- Pull-up for J7
  J8  : inout std_logic;
  J8R : out std_logic; -- Pull-up for J8
  -- R/C Oscillator
  R1  : in std_logic;
  R2  : in std_logic;
  CAP : in std_logic;
  -- Reserved Pins?
  res1: in std_logic;

  -- LED (low is LIT)
  LED : out std_logic;
  -- "Dummy" (dont care) output
  DUMMY : out std_logic
  );
end odd_lvtpp;

architecture Behavioral of odd_lvtpp is

  COMPONENT lvtpp PORT(
    DATA_IN_D1 : IN std_logic;
    CLK_IN_D1 : IN std_logic;
    MCLR_IN_D2 : IN std_logic;
```

```
    PGM_IN_D3 : IN std_logic;
    DATA_INOUT : INOUT std_logic;
    DATA_OUT_P10 : OUT std_logic;
    CLK_OUT : OUT std_logic;
    MCLR_OUT : OUT std_logic;
    PGM_OUT : OUT std_logic);
        END COMPONENT;

  COMPONENT odd_dummy PORT(
    TDI_IN : IN std_logic;
    TDO_IN : IN std_logic;
    TCK_IN : IN std_logic;
    TMS_IN : IN std_logic;
    JTAG_DUMMY : OUT std_logic);
    END COMPONENT;

begin
  --
  Inst_lvtpp: lvtpp PORT MAP(
    DATA_INOUT => J8,
    DATA_IN_D1 => D1,
    DATA_OUT_P10 => P10,
    CLK_OUT => J1,
    CLK_IN_D1 => D1,
    MCLR_OUT => J3,
    MCLR_IN_D2 => D2,
    PGM_OUT => J4,
    PGM_IN_D3 => D3);
  --
  J8R <= '1';

  -- Dummy instance to keep JTAG pins as input
  Inst_odd_dummy: ODD_dummy PORT MAP(
    TDI_IN => D0,
    TDO_IN => D1,
    TCK_IN => D2,
    TMS_IN => P13,
    JTAG_DUMMY => DUMMY);

  -- Make LED Lit always ?
  LED <= '0';

end Behavioral;
```

1.2.10. SP12 Programmer

1.2.11. Atmel/Kanda STK200/STK300

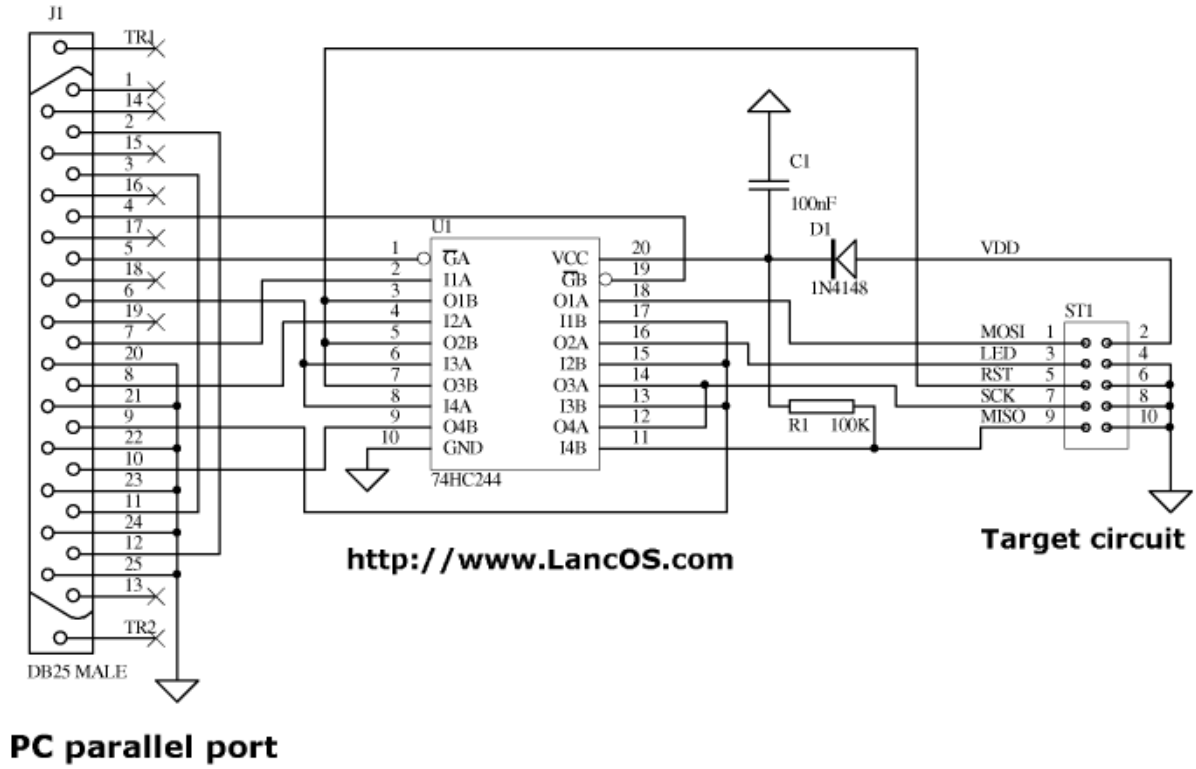

Figure 1.12. STK200 Schematics

Example 1.13. stk200.vhd

```
-- Copyright 2003 Nugis Foundation.
-- STK200/STK300 AVR ISP Programmer, as per www.lancos.com docs
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity stk200 is Port (
  SEN1_IN_D0   : in std_logic;
  SEN2_IN_D1   : in std_logic;
  SEN1_OUT_P12 : out std_logic;
  SEN2_OUT_P11 : out std_logic;
  GA_IN_D3     : in std_logic;
  GB_IN_D2     : in std_logic;
  RST_IN_D7    : in std_logic;
  RST_OUT_J5   : out std_logic;
  SCK_IN_D4    : in std_logic;
  SCK_OUT_J7   : out std_logic;
  LED_IN_D6    : in std_logic;
  LED_OUT_J3   : out std_logic;
  MOSI_IN_D5   : in std_logic;
  MOSI_OUT_J1  : out std_logic;
  MISO_OUT_P10 : out std_logic;
  MISO_IN_J9   : in std_logic);
end stk200;

architecture Behavioral of stk200 is
begin
  -- HC244 A-side
  MOSI_OUT_J1 <= MOSI_IN_D5 when (GA_IN_D3 = '0') else 'Z';
  SCK_OUT_J7 <= SCK_IN_D4 when (GA_IN_D3 = '0') else 'Z';
  LED_OUT_J3 <= LED_IN_D6 when (GA_IN_D3 = '0') else 'Z';
```

```
  -- HC244 B-side
  RST_OUT_J5 <= RST_IN_D7 when (GB_IN_D2 = '0') else 'Z';
  MISO_OUT_P10 <= MISO_IN_J9 when (GB_IN_D2 = '0') else 'Z';
  -- sense loop-backs
  SEN1_OUT_P12 <= SEN1_IN_D0;
  SEN2_OUT_P11 <= SEN2_IN_D1;
end Behavioral;
```

Example 1.14. odd_stk200.vhd

```
-- Copyright 2003 Nugis Foundation
library IEEE;

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

use work.ALL;

entity odd_stk200 is Port (
  -- LPT Port Data (Pin 2..9)
  D0 : in std_logic;
  D1 : in std_logic;
  D2 : in std_logic;
  D3 : in std_logic;
  D4 : in std_logic;
  D5 : in std_logic;
  D6 : in std_logic;
  D7 : in std_logic;
  -- LPT Port Control/Status ( Pn = Pin(n) )
  P10 : out std_logic;
  P11 : out std_logic;
  P12 : out std_logic;
  P13 : in std_logic;
  P14 : in std_logic;
  P15 : in std_logic;
  P16 : in std_logic;
  P17 : in std_logic;
  -- Programming Connector 10-Way IDC
  J1  : out std_logic;
  J3  : out std_logic;
  J4  : in std_logic;
  J5  : out std_logic;
  J6  : in std_logic;
  J7  : out std_logic;
  J7R : in std_logic; -- Pull-up for J7
  J8  : in std_logic;
  J8R : out std_logic; -- Pull-up for J8
  -- R/C Oscillator
  R1  : in std_logic;
  R2  : in std_logic;
  CAP : in std_logic;
  -- Reserved Pins?
  res1: in std_logic;

  -- LED (low is LIT)
  LED : out std_logic;
  -- "Dummy" (dont care) output
  DUMMY : out std_logic
  );
end odd_stk200;

architecture Behavioral of odd_stk200 is

  COMPONENT stk200 PORT(
    SEN1_IN_D0 : IN std_logic;
    SEN2_IN_D1 : IN std_logic;
    SEN1_OUT_P12 : OUT std_logic;
    SEN2_OUT_P11 : OUT std_logic;
    GA_IN_D3 : IN std_logic;
    GB_IN_D2 : IN std_logic;
    RST_IN_D7 : IN std_logic;
```

```
    SCK_IN_D4 : IN std_logic;
    LED_IN_D6 : IN std_logic;
    MOSI_IN_D5 : IN std_logic;
    MISO_IN_J9 : IN std_logic;
    RST_OUT_J5 : OUT std_logic;
    SCK_OUT_J7 : OUT std_logic;
    LED_OUT_J3 : OUT std_logic;
    MOSI_OUT_J1 : OUT std_logic;
    MISO_OUT_P10 : OUT std_logic);
  END COMPONENT;

  COMPONENT odd_dummy PORT(
    TDI_IN : IN std_logic;
    TDO_IN : IN std_logic;
    TCK_IN : IN std_logic;
    TMS_IN : IN std_logic;
    JTAG_DUMMY : OUT std_logic);
    END COMPONENT;

begin
  Inst_stk200: stk200 PORT MAP(
    SEN1_IN_D0 => D0,
    SEN2_IN_D1 => D1,
    SEN1_OUT_P12 => P12,
    SEN2_OUT_P11 => P11,
    GA_IN_D3 => D3,
    GB_IN_D2 => D2,
    RST_IN_D7 => D7,
    RST_OUT_J5 => J5,
    SCK_IN_D4 => D4,
    SCK_OUT_J7 => J7,
    LED_IN_D6 => D6,
    LED_OUT_J3 => J3,
    MOSI_IN_D5 => D5,
    MOSI_OUT_J1 => J1,
    MISO_OUT_P10 => P10,
    MISO_IN_J9 => J8);

  J8R <= '1';

  -- Dummy instance to keep JTAG pins as input
  Inst_odd_dummy: ODD_dummy PORT MAP(
    TDI_IN => D0,
    TDO_IN => D1,
    TCK_IN => D2,
    TMS_IN => P13,
    JTAG_DUMMY => DUMMY);

  -- Make LED Lit always ?
  LED <= '0';

end Behavioral;
```

1.2.12. YAAP (Jesper's AVR Programmer)

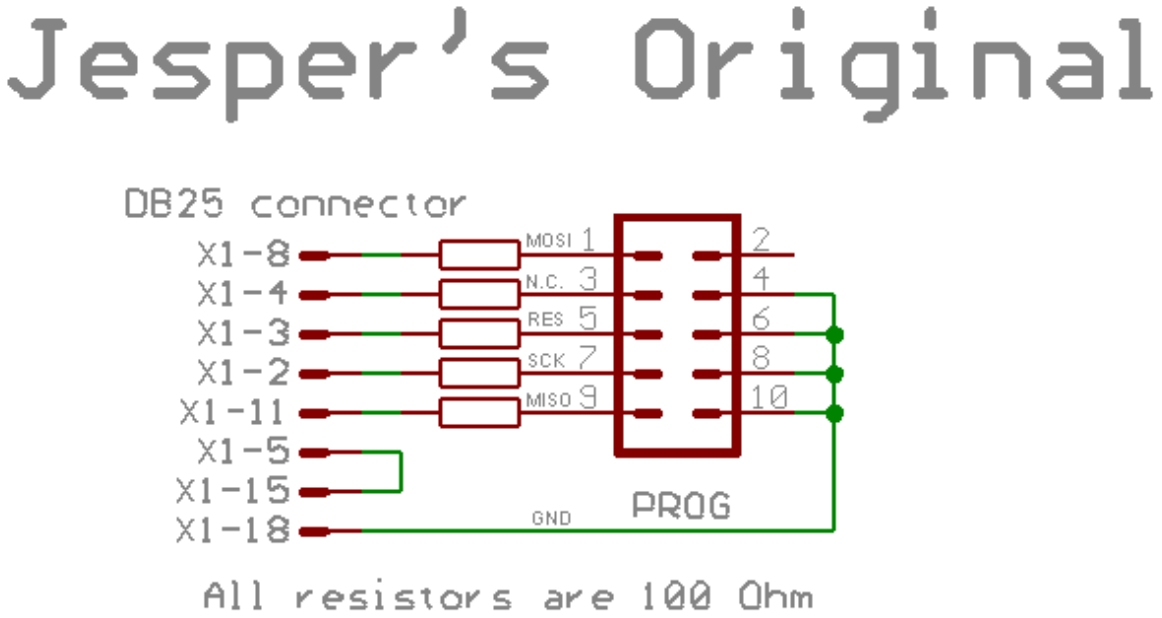

Figure 1.13. Jesper's AVR Programmer

Example 1.15. jesper.vhd

```
-- Copyright 2003 Nugis Foundation.
-- Jespers AVR ISP Programmer
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity jesper is Port (
  SEN1_IN_D3   : in std_logic;
  SEN1_OUT_P15 : out std_logic;
  RST_IN_D1    : in std_logic;
  RST_OUT_J5   : out std_logic;
  SCK_IN_D0    : in std_logic;
  SCK_OUT_J7   : out std_logic;
  MOSI_IN_D6   : in std_logic;
  MOSI_OUT_J1  : out std_logic;
  LED_IN_D2    : in std_logic;
  LED_OUT_J3   : out std_logic;
  MISO_OUT_P11 : out std_logic;
  MISO_IN_J9   : in std_logic);
end jesper;

architecture Behavioral of jesper is
begin
  MOSI_OUT_J1 <= MOSI_IN_D6;
  SCK_OUT_J7 <= SCK_IN_D0;
  RST_OUT_J5 <= RST_IN_D1;
  MISO_OUT_P11 <= MISO_IN_J9;
  -- ? N.C. On jespers design, but is routed?
  LED_OUT_J3 <= LED_IN_D2;
  -- sense loop-back
  SEN1_OUT_P15 <= SEN1_IN_D3;
end Behavioral;
```

Example 1.16. odd_jesper.vhd

```
-- Copyright 2003 Nugis Foundation
library IEEE;

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

use work.ALL;

entity odd_jesper is Port (
  -- LPT Port Data (Pin 2..9)
  D0 : in std_logic;
  D1 : in std_logic;
  D2 : in std_logic;
  D3 : in std_logic;
  D4 : in std_logic;
  D5 : in std_logic;
  D6 : in std_logic;
  D7 : in std_logic;
  -- LPT Port Control/Status ( Pn = Pin(n) )
  P10 : in std_logic;
  P11 : out std_logic;
  P12 : in std_logic;
  P13 : in std_logic;
  P14 : in std_logic;
  P15 : out std_logic;
  P16 : in std_logic;
  P17 : in std_logic;
  -- Programming Connector 10-Way IDC
  J1  : out std_logic;
  J3  : out std_logic;
  J4  : in std_logic;
  J5  : out std_logic;
  J6  : in std_logic;
  J7  : out std_logic;
  J7R : in std_logic; -- Pull-up for J7
  J8  : in std_logic;
  J8R : out std_logic; -- Pull-up for J8
  -- R/C Oscillator
  R1  : in std_logic;
  R2  : in std_logic;
  CAP : in std_logic;
  -- Reserved Pins?
  res1: in std_logic;

  -- LED (low is LIT)
  LED : out std_logic;
  -- "Dummy" (dont care) output
  DUMMY : out std_logic
  );
end odd_jesper;

architecture Behavioral of odd_jesper is

  COMPONENT jesper PORT(
    SEN1_IN_D3 : IN std_logic;
    SEN1_OUT_P15 : OUT std_logic;
    LED_IN_D2 : IN std_logic;
    LED_OUT_J3 : OUT std_logic;
    RST_IN_D1 : IN std_logic;
    SCK_IN_D0 : IN std_logic;
    MOSI_IN_D6 : IN std_logic;
    MISO_IN_J9 : IN std_logic;
    RST_OUT_J5 : OUT std_logic;
    SCK_OUT_J7 : OUT std_logic;
    MOSI_OUT_J1 : OUT std_logic;
    MISO_OUT_P11 : OUT std_logic);
  END COMPONENT;

  COMPONENT odd_dummy PORT(
    TDI_IN : IN std_logic;
    TDO_IN : IN std_logic;
```

```
    TCK_IN : IN std_logic;
    TMS_IN : IN std_logic;
    JTAG_DUMMY : OUT std_logic);
    END COMPONENT;

begin
  Inst_jesper: jesper PORT MAP(
    SEN1_IN_D3 => D3,
    SEN1_OUT_P15 => P15,
    RST_IN_D1 => D1,
    RST_OUT_J5 => J5,
    SCK_IN_D0 => D0,
    SCK_OUT_J7 => J7,
    LED_IN_D2 => D2,
    LED_OUT_J3 => J3,
    MOSI_IN_D6 => D6,
    MOSI_OUT_J1 => J1,
    MISO_OUT_P11 => P11,
    MISO_IN_J9 => J8);

  J8R <= '1';

  -- Dummy instance to keep JTAG pins as input
  Inst_odd_dummy: ODD_dummy PORT MAP(
    TDI_IN => D0,
    TDO_IN => D1,
    TCK_IN => D2,
    TMS_IN => P13,
    JTAG_DUMMY => DUMMY);

  -- Make LED Lit always ?
  LED <= '0';

end Behavioral;
```

1.2.13. Lattice Download Cable

1.2.14. Altera Byteblaster

1.2.15. Xilinx Parallel Download III

1.2.16. Vantis Download Cable

1.2.17. Atmel ATDH2225

Programming adapter for Atmel Configuration memories. Use free CPS software.

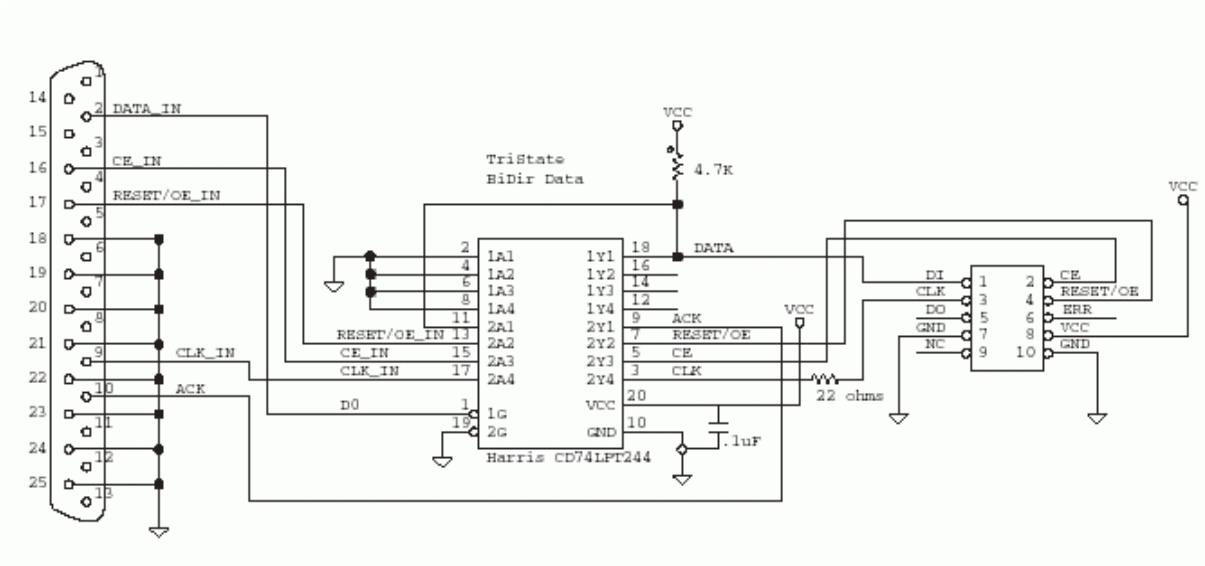

Figure 1.14. ATDH2225

1.2.18. Atmel ATDH2081

Programming adapter for Atmel FPGA(AT6K/AT40K) and FPSLIC(AT94K). Can not be used to program configuration memories.

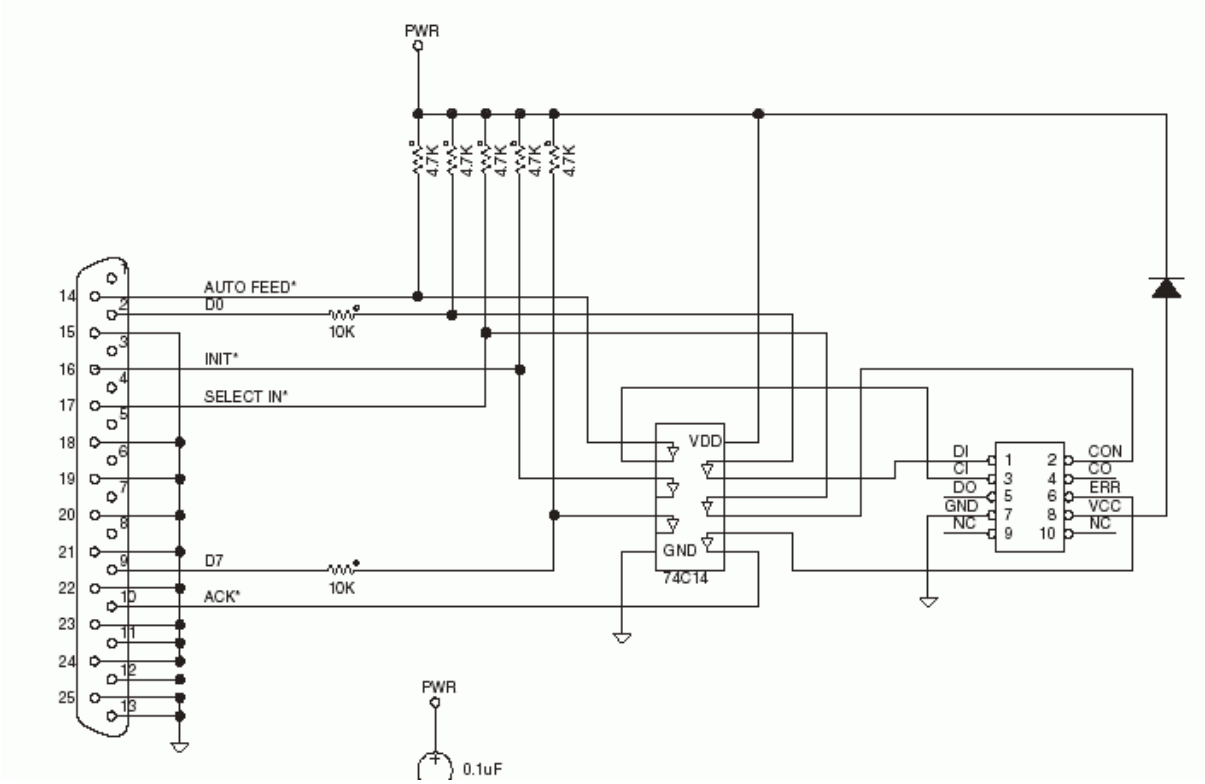

Figure 1.15. ATDH2081

Example 1.17. atdh2081.vhd

```
-- Copyright 2003 Nugis Foundation.
-- Based on doc2315.pdf from www.atmel.com
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity atdh2081 is Port (
  ERR_IN_J6 : in std_logic;
  CON_OUT_J2 : out std_logic;
  DI_OUT_J1 : out std_logic;
  CI_OUT_J3 : out std_logic;
  ERR_OUT_P10 : out std_logic;
  CON_IN_P17 : in std_logic;
  DI_IN_D0 : in std_logic;
  CI_IN_P14 : in std_logic);
end atdh2081;

architecture Behavioral of atdh2081 is
begin
  -- ATDH2081 direct connections over HC14 (inverter)
  ERR_OUT_P10 <= not ERR_IN_J6;
  CON_OUT_J2 <= not CON_IN_P17;
  DI_OUT_J1 <= not DI_IN_D0;
  CI_OUT_J3 <= not CI_IN_P14;
end Behavioral;
```

Example 1.18. odd_atdh2081.vhd

```
-- Copyright 2003 Nugis Foundation
library IEEE;

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

use work.ALL;

entity odd_atdh2081 is Port (
  D0 : in std_logic;
  D1 : in std_logic;
  D2 : in std_logic;

  P10 : out std_logic;
  P13 : in std_logic;
  P14 : in std_logic;
  P16 : in std_logic;
  P17 : in std_logic;
  -- Programming Connector
  J1 : out std_logic;
  J3 : out std_logic;
  J4 : out std_logic;
  J8 : in std_logic;
  -- LED
  LED : out std_logic;
  -- "Dummy" (dont care) output
  DUMMY : out std_logic
  );
end odd_atdh2081;

architecture Behavioral of odd_atdh2081 is

    COMPONENT atdh2081 PORT(
      ERR_IN_J6 : IN std_logic;
      CON_IN_P17 : IN std_logic;
      DI_IN_D0 : IN std_logic;
      CI_IN_P14 : IN std_logic;
```

```
      ERR_OUT_P10 : OUT std_logic;
      CON_OUT_J2 : OUT std_logic;
      DI_OUT_J1 : OUT std_logic;
      CI_OUT_J3 : OUT std_logic
                );
    END COMPONENT;

    COMPONENT odd_dummy PORT(
      TDI_IN : IN std_logic;
      TDO_IN : IN std_logic;
      TCK_IN : IN std_logic;
      TMS_IN : IN std_logic;
      JTAG_DUMMY : OUT std_logic
      );
    END COMPONENT;

begin
  -- ATDH2081
  Inst_atdh2081: atdh2081 PORT MAP(
    ERR_IN_J6 => J8,      -- ERR(6) = J8 (optional pull-up possible)
    ERR_OUT_P10 => P10,
    CON_OUT_J2 => J4,     -- CON(2) = J4
    CON_IN_P17 => P17,
    DI_OUT_J1 => J1,      -- DI(1) = J1
    DI_IN_D0 => D0,
    CI_OUT_J3 => J3,      -- CI(3) = J3
    CI_IN_P14 => P14
  );

  -- Dummy instance to keep JTAG pins as input
  Inst_odd_dummy: ODD_dummy PORT MAP(
    TDI_IN => D0,
    TDO_IN => D1,
    TCK_IN => D2,
    TMS_IN => P13,
    JTAG_DUMMY => DUMMY
  );

  -- Make LED Lit always ?
  LED <= '0';

end Behavioral;
```

1.2.19. MSP430-JTAG

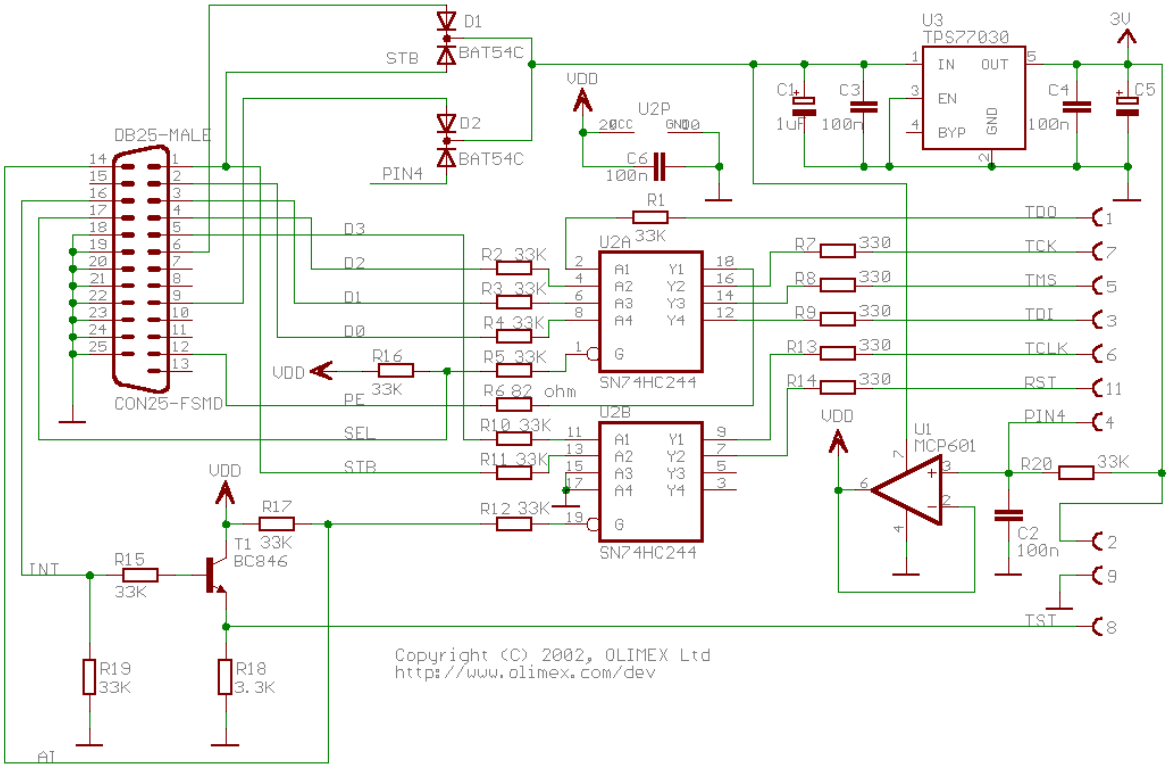

Figure 1.16. MSP430 Schematics

Example 1.19. msp430jtag.vhd

```
-- Copyright 2003 Nugis Foundation.
-- MSP430 JTAG Interface (TI FET Tool compatible)
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity msp430jtag is Port (
  GA_IN_P17   : in std_logic;
  GB_IN_P14   : in std_logic;
  TCK_OUT_J7  : out std_logic;
  TDI_OUT_J3  : out std_logic;
  TDO_IN_J1   : in std_logic;
  TMS_OUT_J5  : out std_logic;
  TCLK_OUT_J6 : out std_logic;
  TST_OUT_J8  : out std_logic;
  RST_OUT_J11 : out std_logic;
  TCK_IN_D2   : in std_logic;
  TDI_IN_D0   : in std_logic;
  TDO_OUT_P12 : out std_logic;
  TMS_IN_D1   : in std_logic;
  TCLK_IN_D3  : in std_logic;
  TST_IN_P16  : in std_logic;
  RST_IN_P1   : in std_logic);

end msp430jtag;

architecture Behavioral of msp430jtag is
begin
  -- HC244 a-side
  TCK_OUT_J7 <= TCK_IN_D2 when (GA_IN_P17 = '0') else 'Z';
  TDI_OUT_J3 <= TDI_IN_D0 when (GA_IN_P17 = '0') else 'Z';
```

```
  TMS_OUT_J5 <= TMS_IN_D1 when (GA_IN_P17 = '0') else 'Z';
  TDO_OUT_P12 <= TDO_IN_J1 when (GA_IN_P17 = '0') else 'Z';
  -- HC244 b-side
  RST_OUT_J11 <= not RST_IN_P1 when (GB_IN_P14 = '0') else 'Z';
  TCLK_OUT_J6 <= not TCLK_IN_D3 when (GB_IN_P14 = '0') else 'Z';
  --
  TST_OUT_J8 <= TST_IN_P16;
end Behavioral;
```

Example 1.20. odd_msp430jtag.vhd

```
-- Copyright 2003 Nugis Foundation
library IEEE;

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

use work.ALL;

entity odd_msp430jtag is Port (
  -- LPT Port Data (Pin 2..9)
  D0 : in std_logic;
  D1 : in std_logic;
  D2 : in std_logic;
  D3 : in std_logic;
  D4 : in std_logic;
  D5 : in std_logic;
  D6 : in std_logic;
  D7 : in std_logic;
  -- LPT Port Control/Status ( Pn = Pin(n) )
  P1  : in std_logic;
  P10 : in std_logic;
  P11 : in std_logic;
  P12 : out std_logic;
  P13 : in std_logic;
  P14 : in std_logic;
  P15 : in std_logic;
  P16 : in std_logic;
  P17 : in std_logic;
  -- Programming Connector 10-Way IDC
  J1  : out std_logic;
  J3  : out std_logic;
  J4  : out std_logic;
  J5  : out std_logic;
  J6  : out std_logic;
  J7  : out std_logic;
  J7R : in std_logic; -- Pull-up for J7
  J8  : in std_logic;
  J8R : in std_logic; -- Pull-up for J8
  -- R/C Oscillator
  R1  : in std_logic;
  R2  : in std_logic;
  CAP : in std_logic;
  -- Reserved Pins?
  res1: in std_logic;

  -- LED (low is LIT)
  LED : out std_logic;
  -- "Dummy" (dont care) output
  DUMMY : out std_logic
  );
end odd_msp430jtag;

architecture Behavioral of odd_msp430jtag is

  COMPONENT msp430jtag PORT(
    GA_IN_P17 : IN std_logic;
    GB_IN_P14 : IN std_logic;
    TDO_IN_J1 : IN std_logic;
    TCK_IN_D2 : IN std_logic;
    TDI_IN_D0 : IN std_logic;
```

```
    TMS_IN_D1 : IN std_logic;
    TCLK_IN_D3 : IN std_logic;
    TST_IN_P16 : IN std_logic;
    RST_IN_P1 : IN std_logic;
    TCK_OUT_J7 : OUT std_logic;
    TDI_OUT_J3 : OUT std_logic;
    TMS_OUT_J5 : OUT std_logic;
    TCLK_OUT_J6 : OUT std_logic;
    TST_OUT_J8 : OUT std_logic;
    RST_OUT_J11 : OUT std_logic;
    TDO_OUT_P12 : OUT std_logic);
   END COMPONENT;

  COMPONENT odd_dummy PORT(
    TDI_IN : IN std_logic;
    TDO_IN : IN std_logic;
    TCK_IN : IN std_logic;
    TMS_IN : IN std_logic;
    JTAG_DUMMY : OUT std_logic);
  END COMPONENT;

begin
  Inst_msp430jtag: msp430jtag PORT MAP(
    GA_IN_P17 => P17,
    GB_IN_P14 => P14,
    TCK_OUT_J7 => J7,
    TDI_OUT_J3 => J3,
    TDO_IN_J1 => J8,
    TMS_OUT_J5 => J5,
    TCLK_OUT_J6 => J6,
    TST_OUT_J8 => J1,
    RST_OUT_J11 => J4,
    TCK_IN_D2 => D2,
    TDI_IN_D0 => D0,
    TDO_OUT_P12 => P12,
    TMS_IN_D1 => D1,
    TCLK_IN_D3 => D3,
    TST_IN_P16 => P16,
    RST_IN_P1 => P1);

  -- Dummy instance to keep JTAG pins as input
  Inst_odd_dummy: ODD_dummy PORT MAP(
    TDI_IN => D0,
    TDO_IN => D1,
    TCK_IN => D2,
    TMS_IN => P13,
    JTAG_DUMMY => DUMMY);

  -- Make LED Lit always ?
  LED <= '0';

end Behavioral;
```

1.2.20. BDM PD

BDM PD (Public Domain) Interface

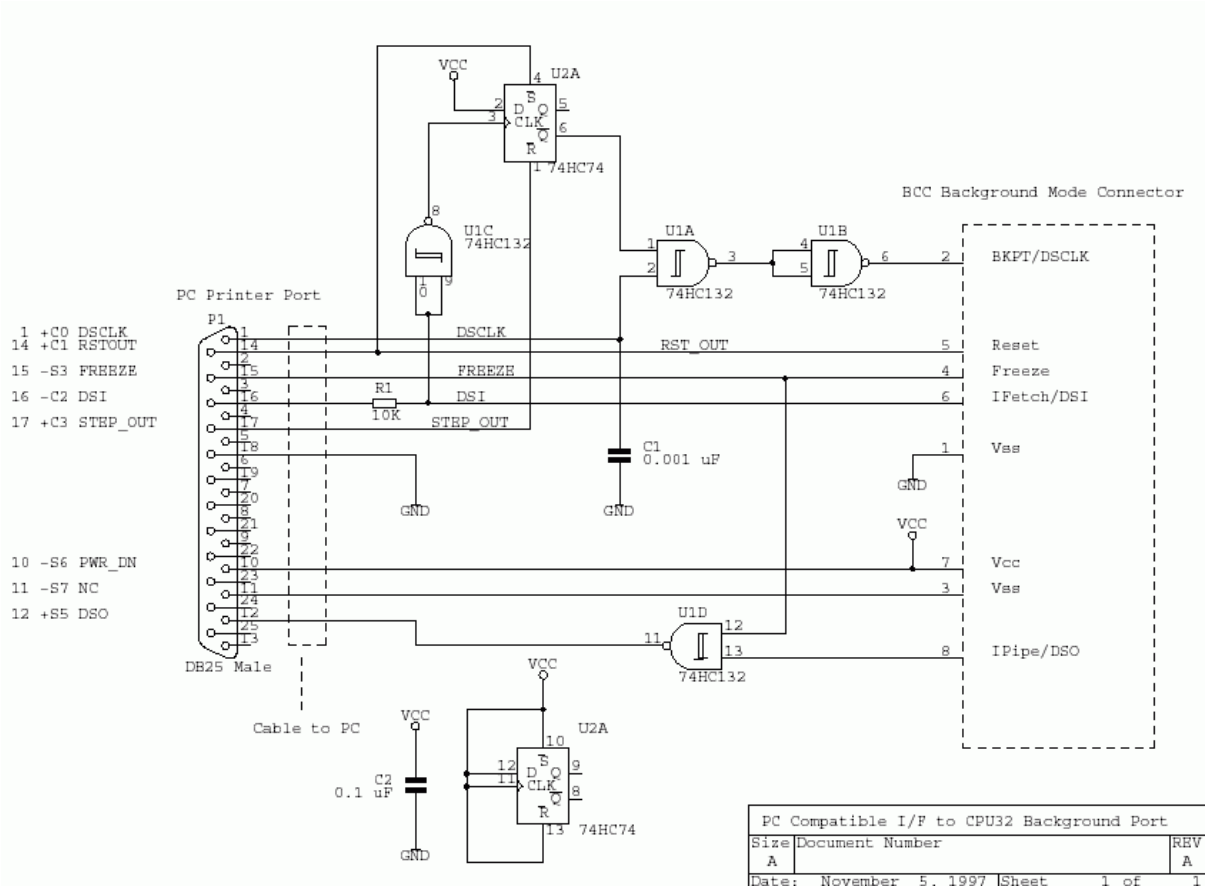

Figure 1.17. BDM PD Schematics

Chapter 2. Similar Projects

2.1. Similar Projects

2.1.1. Amontec's Chameleon POD

Figure 2.1. Chameleon POD

- U1: A Xilinx Coolrunner XCR3128XL-VQ100 CPLD (programmable logic, 5V and 3.3V tolerant I/O pins)
- Q1: A 32MHz Crystal Oscillator
- P1: A D-Sub 25 pin male Connector (computer side)
- P2: A D-Sub 25 pin female Connector (for user side and for power source)
- P3: A power Jack Connector
- S1: A Slide Switch allowing the mode selection ('configure' mode or 'use' mode)
- LED1: A red LED for custom uses
- LED2: A yellow LED for configuration mode signalization
- LED3: A green LED for power signalization
- A DC-DC Linear Regulator allowing the Chameleon POD to be used in different power level environments

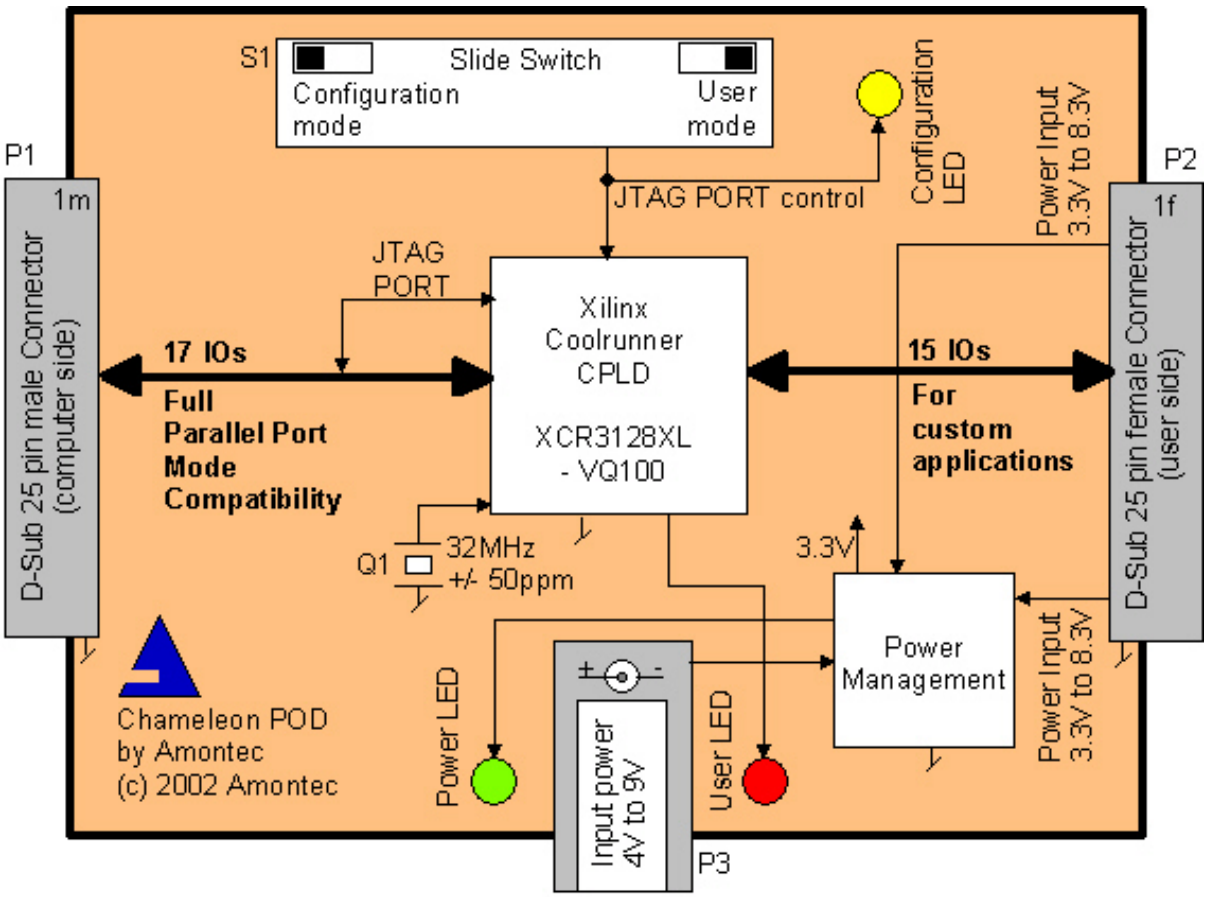

Figure 2.2. Chameleon Diagram

2.1.2. BDM4GDB

Project at sourceforge - status: DEAD. Original BDM Project done by frank.przybylski@vas-gmbh.de ? Just including here as example of almost similar hardware to ODD.

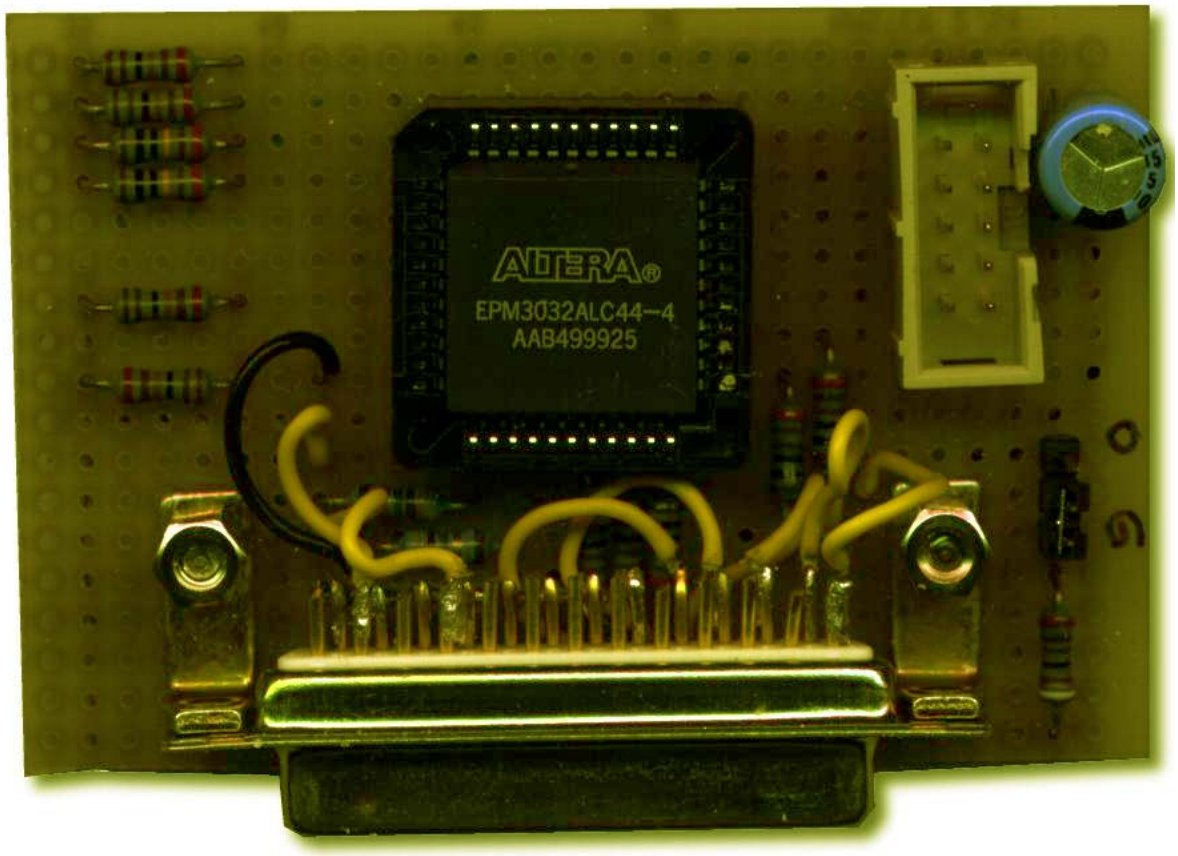

Figure 2.3. BDM_Altera

2.1.3. ULD1016

"Universal Lattice Dongle" - grandpa of ODD, based on ispLSI1016 a 64 Macrocell PLD from Lattice Semiconductor. First configurations where made with PDS software, later with ispLEVER. One prototype exists.

2.1.4. ISP Master 2.0

Switch selectable to emulate a limited number of interfaces.

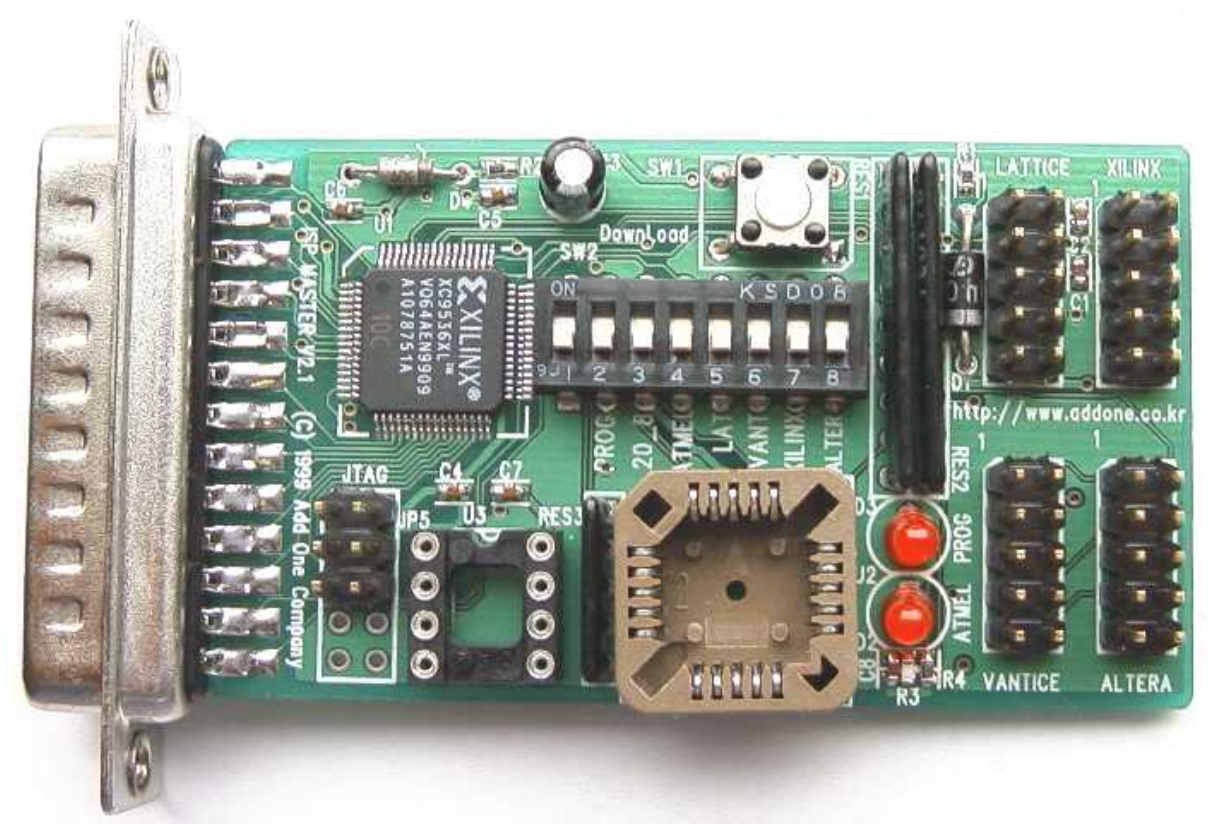

Figure 2.4. ISP Master 2.0

2.1.5. ISP Master 3.0

Enhanced version of ISP Master 2.0

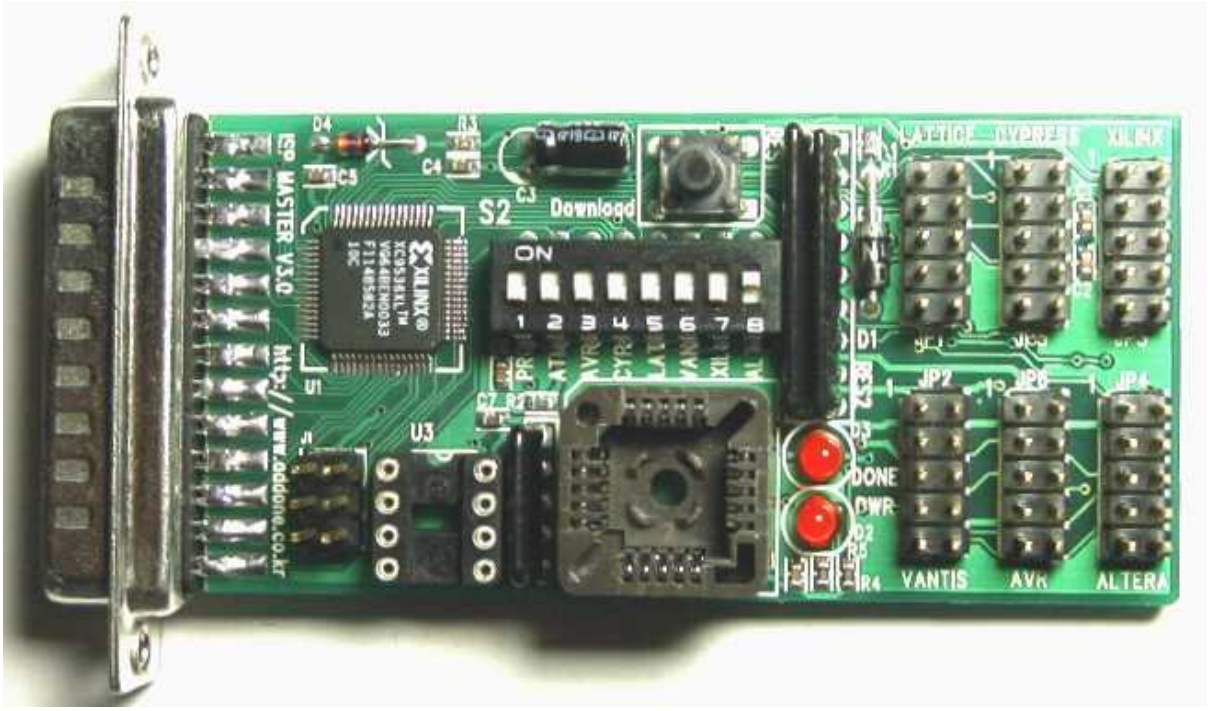

Figure 2.5. ISP Master 3.0