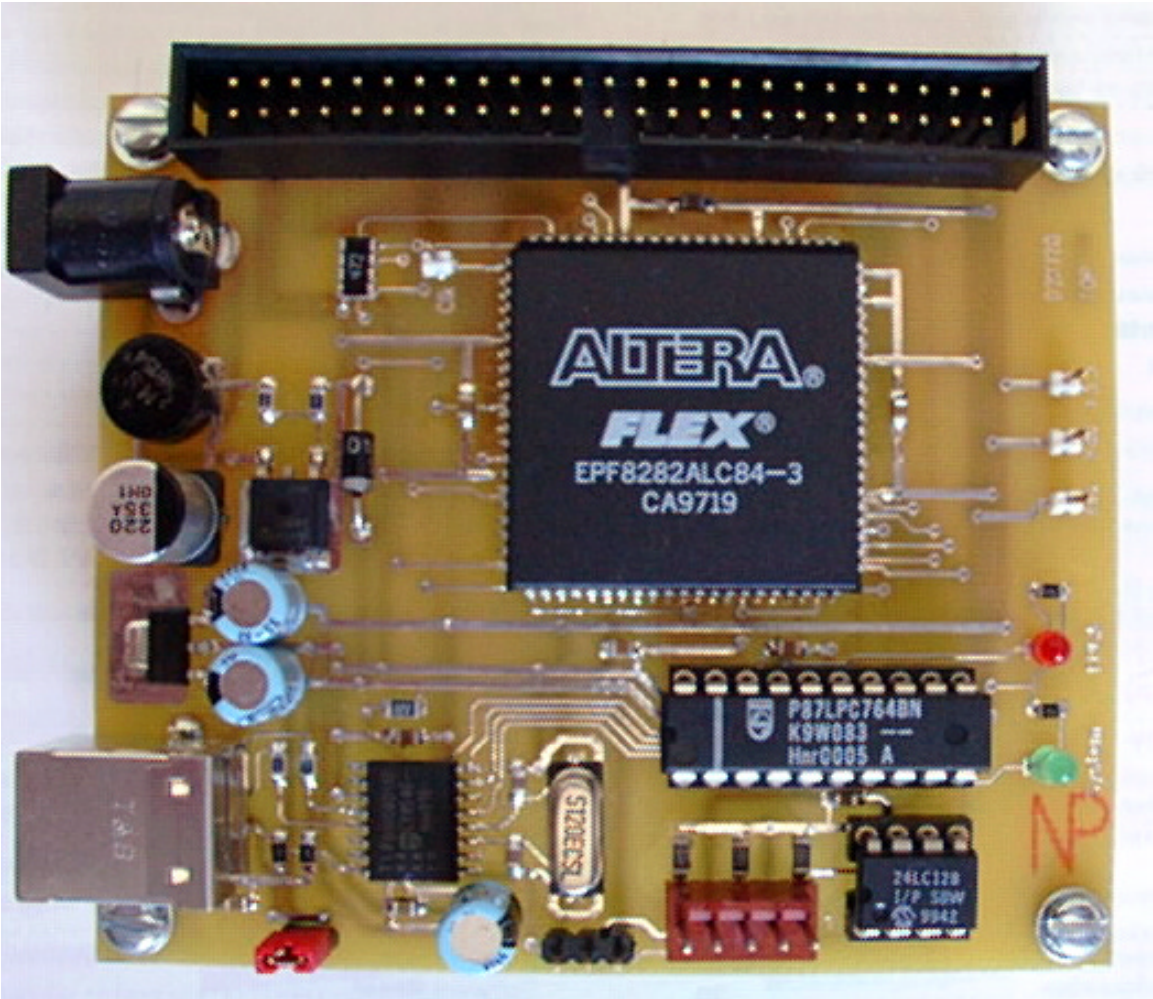
FPGA on a Rope

USB to FPGA



Circuit Cellar Design 2K Contest

Entry #D2K128

Overview

The “FPGA on a Rope” board and device driver provide a simple method to quickly configure and control a user definable FPGA connected to a Windows 98 P.C. through a USB port.

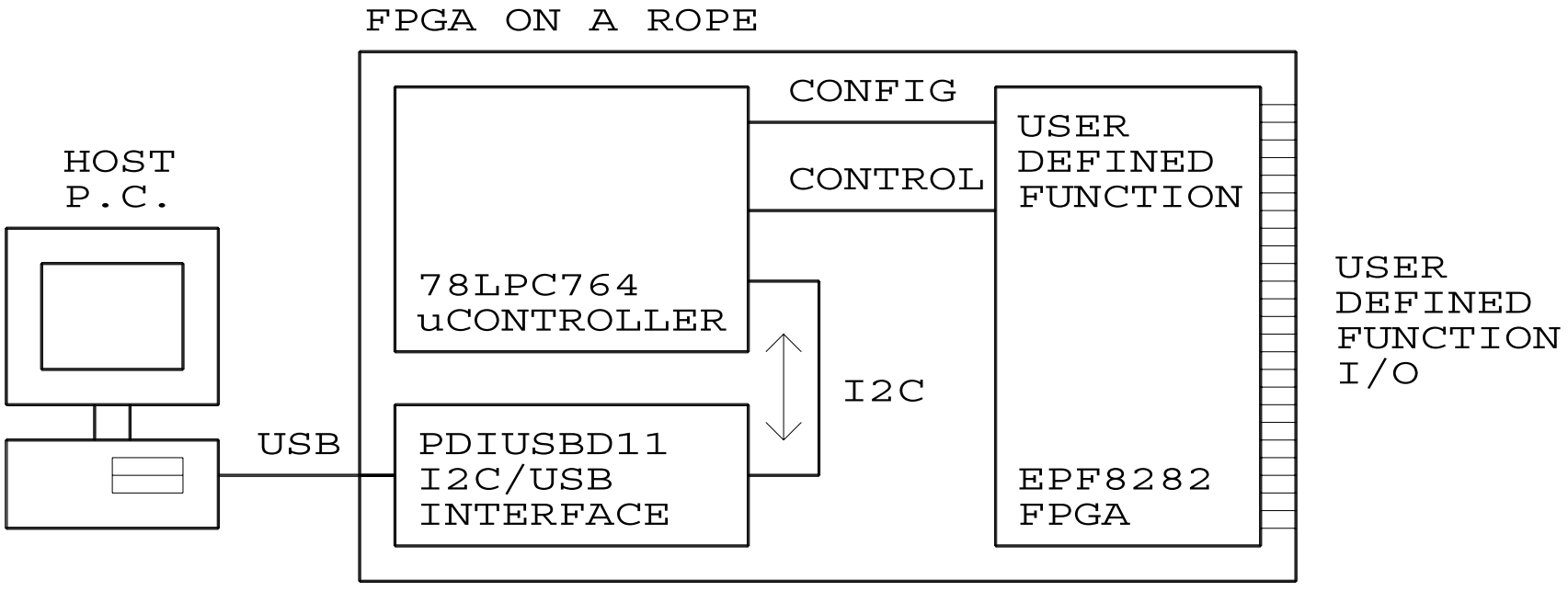
The design consists of the USBFPGA hardware (printed circuit board), and a Windows 98 compatible WDM device driver (USBFPGA.Sys).

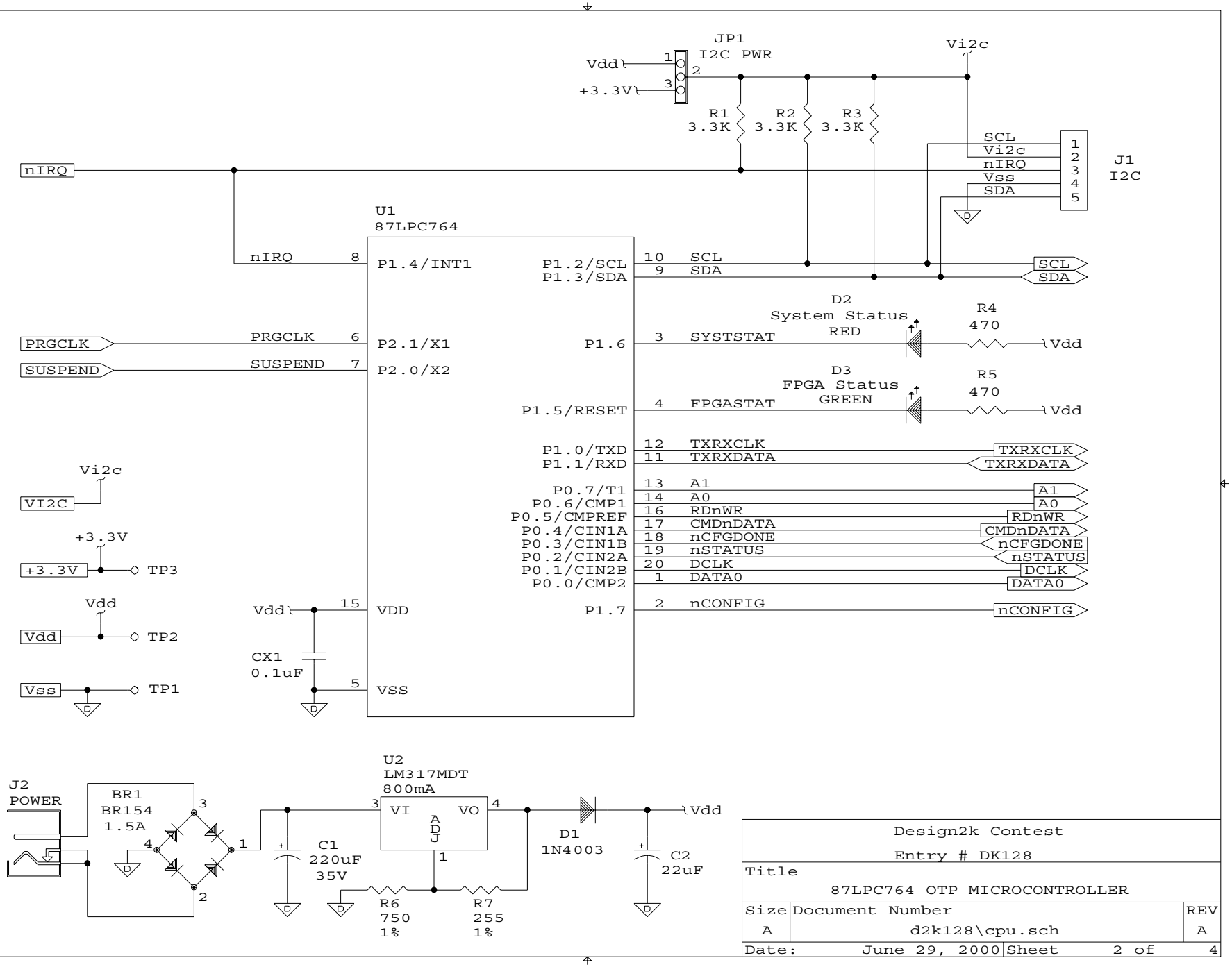
A download application is included (USBFPGA.Exe) which provides an interface for downloading the users FPGA definition files.

An example application and FPGA template design are included to demonstrate the control and status functions of the USBFPGA device. These functions allow the user to develop custom applications that communicate with their defined FPGA hardware via the USB port.

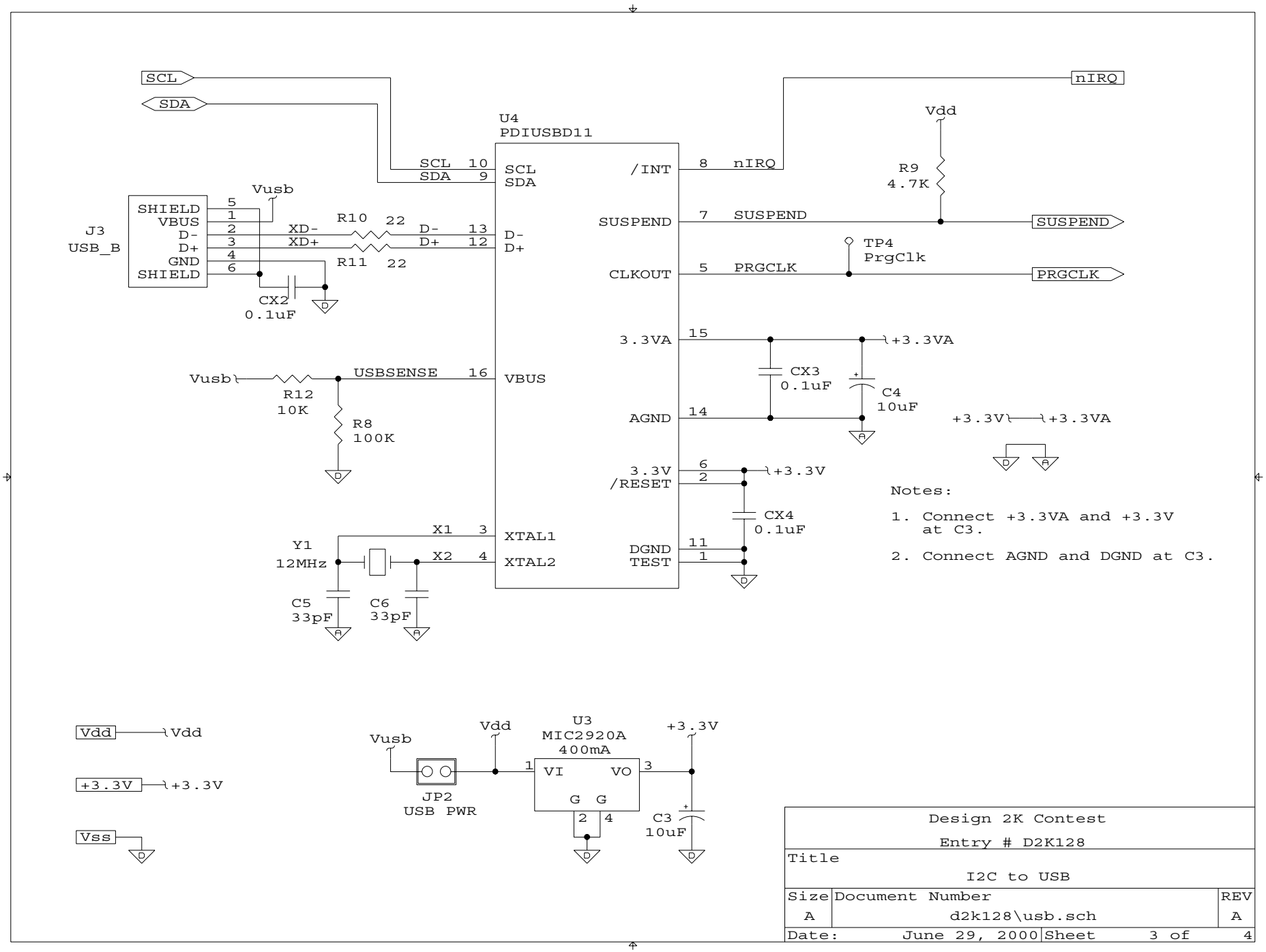
Usage

The USBFPGA is primarily intended for use by technical people such as hardware engineers, software engineers, and electronics hobbyist.

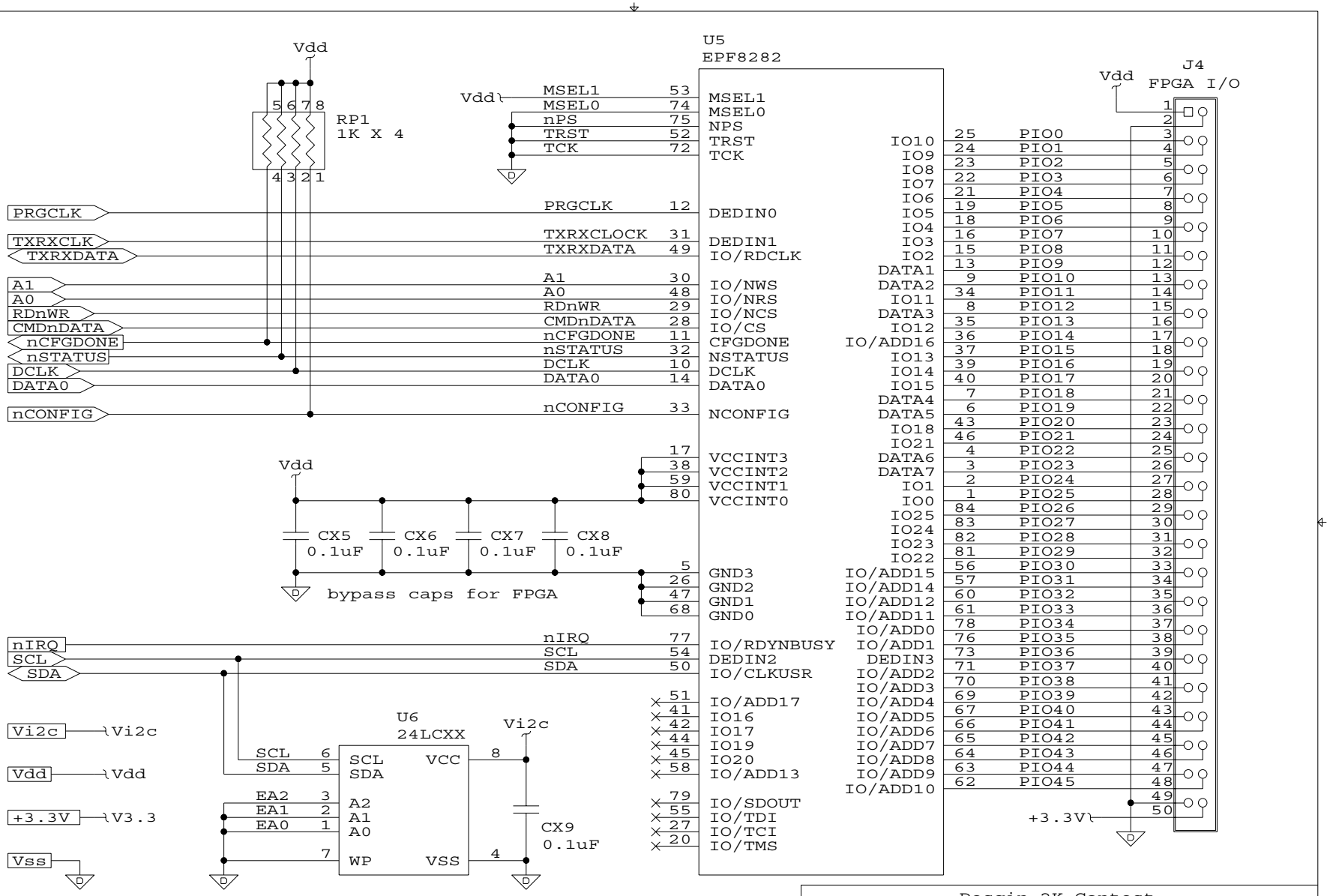

FPGA ON A ROPE
HOST
P.C.
CONFIG
CONTROL
USER
DEFINED
FUNCTION
78LPC764
uCONTROLLER
I2C
USB
PDIUSBD11
I2C/USB
INTERFACE
EPF8282
FPGA
USER
DEFINED
FUNCTION
I/O



Design2k Contest		
Entry # DK128		
Title: 87LPC764 OTP MICROCONTROLLER		
Size: A	Document Number: d2k128\cpu.sch	REV: A
Date: June 29, 2000	Sheet 2 of 4	



Design 2K Contest Entry # D2K128		
Title I2C to USB		
Size A	Document Number d2k128\usb.sch	REV A
Date: June 29, 2000	Sheet 3 of 4	



Desgin 2K Contest		
Entry # D2K128		
Title		
I2C eeProm and FPGA		
Size	Document Number	REV
A	d2k128\memfpga.sch	A
Date: June 29, 2000	Sheet 4 of 4	

Components

The heart of the circuit is a Philips 87LPC764 micro-controller. This component is an 8051 derivative featuring 4KB of one-time programmable ROM, 64 bytes of RAM, and an assortment of on-chip resources (timers, UART, I^2C, reset, etc.). All of the above features are shoe-horned into a 20 pin package. Because the part implements an 8051 core, a wide variety of development tools are available. The Philips 87LPC764 micro-controller has a wide operating voltage range (2.7V to 6V) which allows it to operate from Vusb (note: Section 7.2.2 of the USB specification shows this voltage can be as low as 4.375V), far below the minimum voltage required by many 5V parts. The datasheet for this device is available at http://www.phillipssemiconductor.com/pip/P87LPC764FD.

The connection to the host computer is achieved with a Philips PDIUSBD11 USB interface. This component features an I^2C control interface requiring only three signals from the micro-controller. The device supports full speed USB connections (12Mbps), includes memory buffers for the USB transfers, and has an integrated SIE (serial interface engine) to handle the significant transaction protocol and speed requirements of USB. A SoftConnect (Philips Trademark) feature provides support for disconnection/connection signaling to the USB via software control, and the programmable clock output allows us to keep the crystal count (for the design) down to one (12Mhz). The datasheet for the PDIUSBD11 is available at http://www-us.semiconductors.com/usb/products/interface/pdiusbd11/.

The circuit includes a National Semiconductor LM317 voltage regulator for supplying +5V when the board is operated in self powered mode. In self powered mode, the board can operate from a wide range of A.C. or D.C. input power.

The circuit includes a Micrel 2920A-3.3 voltage regulator for supplying +3.3V to the PDIUSBD11 (USB interface I.C.) when the circuit is operating from USB power or when +5V is supplied via either the I^2C connector or the onboard power supply. The regulator has a low dropout voltage of only 370mV at 250mA, burns a mere 140uA of quiescent current, and can source 400mA. The datasheet for the 2920A-3.3 is available at http://www.micrel.com/product-info/products/mic2920a.html.

Power Options

The circuitry supports eight jumper-configurable, power options:

1. Powered by USB, isolated from I^2C power (I^2C power used only for I^2C signal pull-up resistors).
2. Powered by USB, sourcing Vusb to I^2C bus.
3. Powered by USB, sourcing +3.3V to I^2C bus.

4. Powered by +5V supplied via I^2C bus connector, isolated from USB bus power.
5. Powered by +3.3V supplied via I^2C bus connector, isolated from USB bus power.
6. Powered by onboard power supply, isolated from USB and I2C (I^2C power used only for I^2C signal pull-up resistors).
7. Powered by onboard power supply, isolated from USB, sourcing +5V to I2C.
8. Powered by onboard power supply, isolated from USB, sourcing +3.3V to I2C.

Initialization

At power up, the micro-controller initializes the USB and I^2C interfaces.

The processor is clocked by the PRGCLK (programmable clock) signal which is output from the PDIUSBD11. This signal is derived from a 48Mhz clock passed through a programmable divider and defaults to 4Mhz at power up. The divide value for the clock is changed from the default of 12 (48Mhz/12 = 4Mhz) to 3 (48Mhz/3 = 16Mhz) effectively shifting the processor into high gear.

Enumeration

After initialization, the firmware enters a forever loop where it waits for events to occur.

The first event (actually a whole series of events) to occur is USB enumeration.

The USB bus is designed to be plug-and-play compatible, requiring a method for detecting and initializing new devices, this method is called “enumeration”.

There are several situations that can result in enumeration (power up, plug-in, reset, etc.) but in this design they are all handled (from a software perspective) as if the device were just plugged in to the USB bus and the computer just detected its presence.

When the host computer detects our device “plug-in” it will initiate a series of USB transactions with the intent of discovering what our device is. The information it obtains from our device during enumeration allows the host computer to determine what drivers it should load to support our device and what the communication features of our device are (things like maximum packet sizes, power requirements, etc.). During the enumeration process, the host assigns the device a unique address (from 1 to 127), all further communications between the host and the device will use the assigned address.

The USB enumeration process is detailed in chapter 9 of the USB specification which is available at http://www.usb.org/developers/data/usb_20.zip

Operation

After enumeration, the device is ready to perform it's intended function as an FPGA development platform.

The user develops a definition file for the FPGA using the Altera Max+plus II software. This software is available for free download on the Altera Web site, and supports the EPF8282 part in the baseline (free) version. A template design file is included with the USBFPGA to provide the user with a guide for integrating the USBFPGA control functions (if desired). The control functions provide the user the ability to read and write to the defined FPGA via the USB connection.

The firmware, device driver, and download utility all work together to provide the ability to configure the FPGA with the users definition file. The user simply selects the desired definition file from within the download utility, and the FPGA is loaded (configured) via the USB connection. The download utility also provides for the FPGA definition file to be stored in the EEPROM on board the USBFPGA for automatic configuration of the FPGA at power-up. This gives the USBFPGA the ability to operate when detached from the USB cable.

If the user implemented the USBFPGA control functions in the FPGA definition, then the USBFPGA can be used in it's secondary function (after configuration) as a control interface to the defined FPGA. The device driver provides support for calls to Read and Write registers which exist in the defined FPGA, providing an application specific control interface to the users defined FPGA function.

Access to the user defined I/O pins of the FPGA is provided via a fifty pin header.

Source Code (firmware)

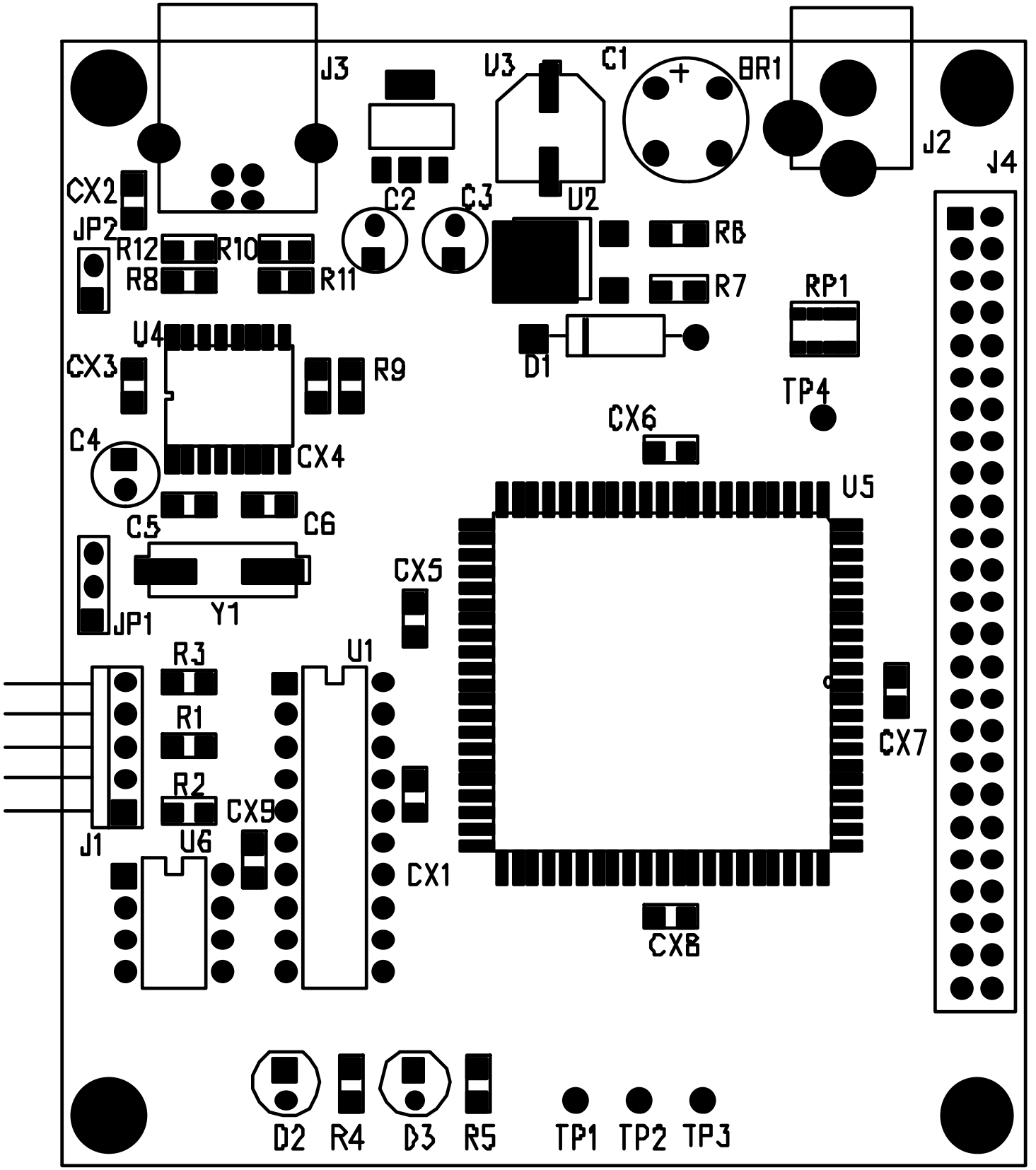
The source code files for the firmware are contained in the file “firmware.zip”.

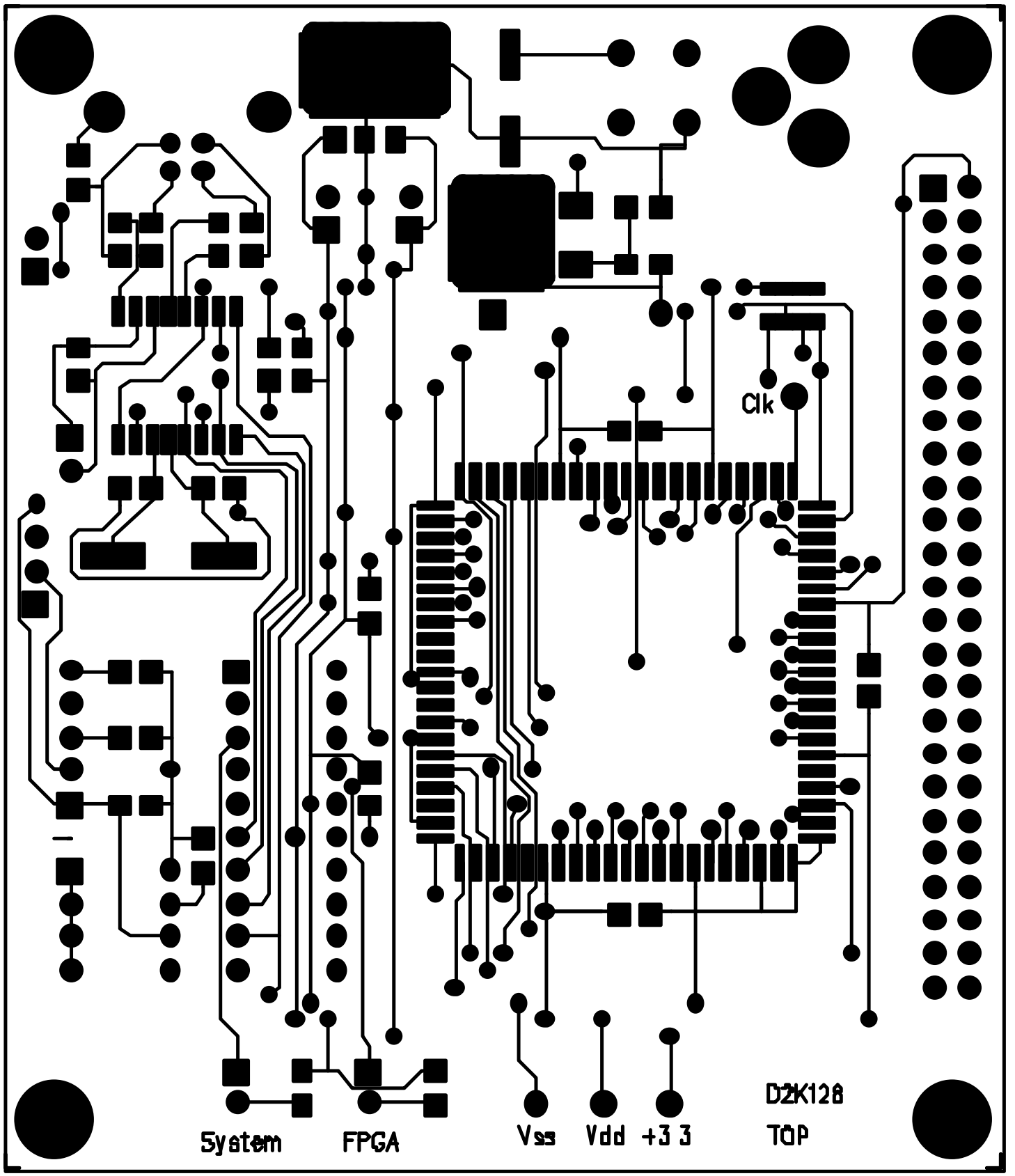
Source Code (Windows WDM device driver)

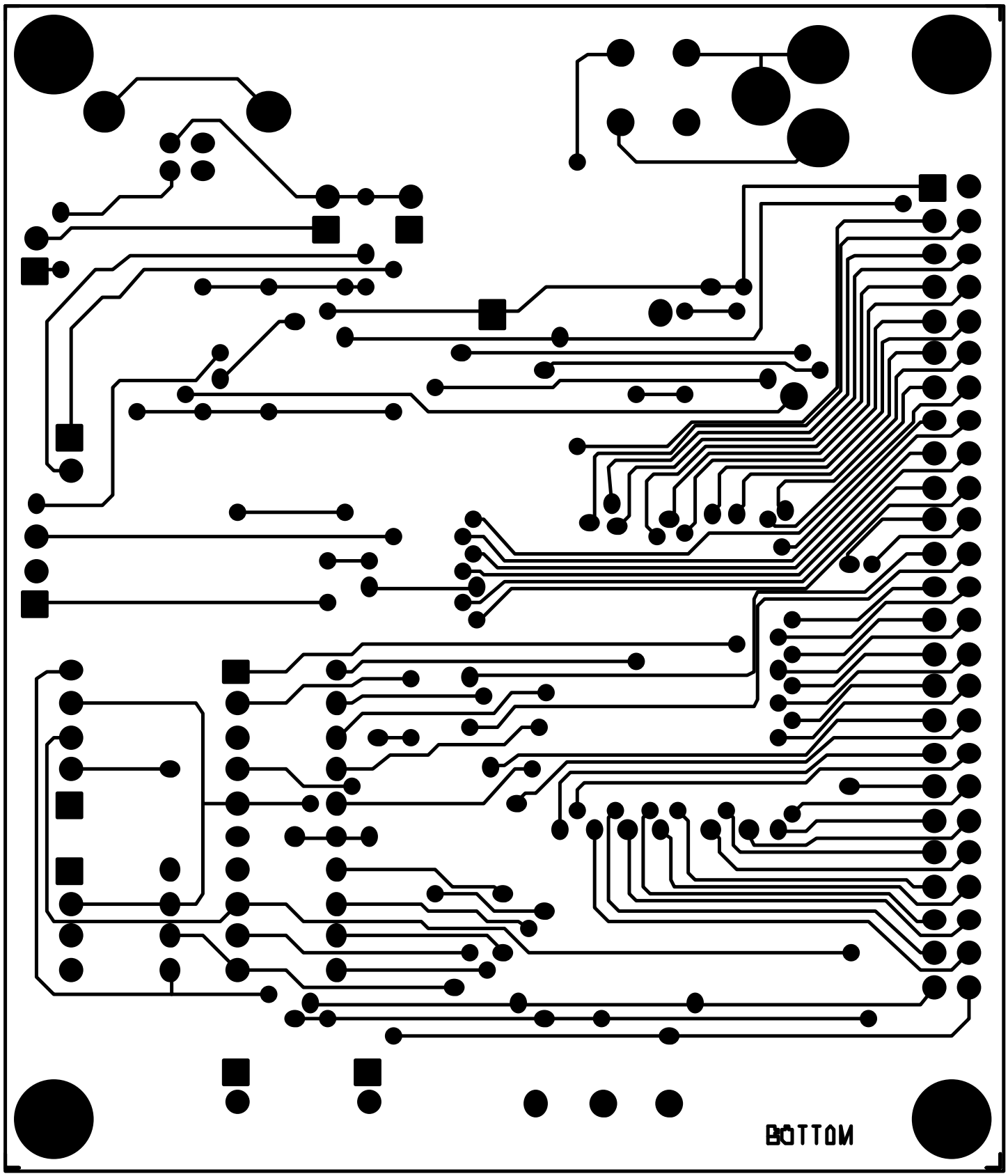
The source code files for the Windows WDM device driver are contained in the file “driver.zip”.

Bill of Materials

Item	Quantity	Reference	Part
1	1	BR1	BR154, 1.5A
2	9	CX1,CX2,CX3,CX4,CX5,CX6, CX7,CX8,CX9	0.1uF
3	1	C1	220uF, 35V
4	1	C2	22uF
5	2	C3,C4	10uF
6	2	C5,C6	33pF
7	1	D1	1N4003
8	1	D2	LED, RED
9	1	D3	LED, GREEN
10	1	JP1	JUMPER 3 PIN
11	1	JP2	JUMPER 2 PIN
12	1	J1	CON5 RA MALE
13	1	J2	POWERJACK
14	1	J3	USB Type B,
15	1	J4	IDC HEADER 50, FPGA I/O
16	1	RP1	1K X 4
17	3	R1,R2,R3	3.3K
18	2	R4,R5	470
19	1	R6	750, 1%
20	1	R7	255, 1%
21	1	R8	100K
22	1	R9	4.7K
23	2	R10,R11	22
24	1	R12	10K
25	3	TP1,TP2,TP3,TP4	TESTPOINT
26	1	U1	87LPC764,
27	1	U2	LM317MDT, 800mA
28	1	U3	MIC2920A-3.3BS, 400mA
29	1	U4	PDIUSBD11
30	1	U5	EPF8282
31	1	U6	24LCXX
32	1	Y1	12MHz


J3
U3
C1
BR1
J2
J4
CX2
C2
C3
U2
JP2
R12
R10
R6
R8
R11
R7
RP1
U4
CX3
R9
D1
TP4
CX6
C4
CX4
U5
C5
C6
CX5
Y1
JP1
U1
R3
R1
CX7
R2
CX9
J1
U6
CX1
CX8
D2
R4
D3
R5
TP1
TP2
TP3


Clk
System
FPGA
Vss
Vdd
+3 3
D2K128
TOP


BOTTOM

Download Utility

The download application “UsbFpga.exe” provides a simple interface for downloading the FPGA definition file.

Download Utility Source Code (Visual Basic)

The source code files for the UsbFpga.exe Visual Basic application are contained in the file usbfpga.zip.