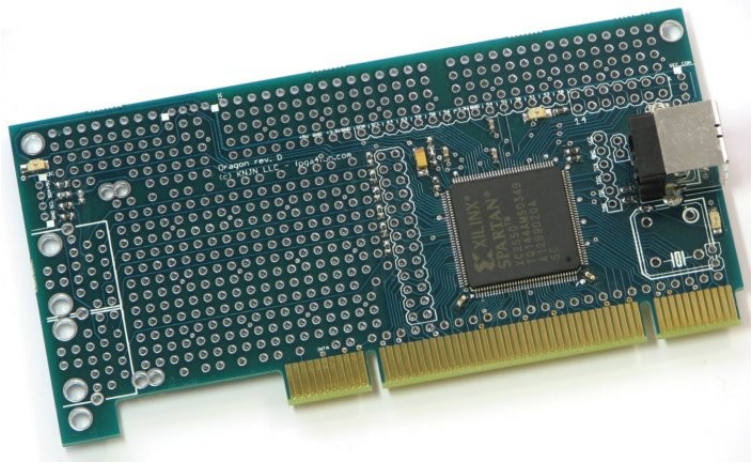
KNJN PCI FPGA Dragon development board

© KNJN LLC

http://www.knjn.com/

This document applies to the following board.

- Dragon rev. D & E



Document last revision on **October 26, 2026**

Table of Contents

1 Welcome 4
1.1 This guide 4
1.2 USB driver installation 4
1.3 Power connection 4
1.4 Software tools 4
1.5 Linux support 4

2 FPGA configuration 5
2.1 FPGA configuration using USB 5
2.2 FPGA configuration using JTAG 5
2.3 Boot-PROM 5

3 FPGA projects with Xilinx's ISE 6

4 Pins and headers 7
4.1 FPGA main pins 7
4.2 FPGA IOs 7

5 USB 8
5.1 USB controller 8
5.2 USB data interface 8
5.3 USB protocol 8

6 USB examples 9
6.1 USB IO8 example 9
6.2 USB Text LCD example 9
6.3 USB Register banks example 9

7 PCI 10
7.1 PCI 10
7.2 PCI and USB 10
7.3 PCI power 10
7.4 PCI BIOS 10

8 PCI designs (non plug-and-play) 11
8.1 PCI designs 11
8.2 PCI_IORAM 11
8.3 PCI_IORAM_LCD 11
8.4 PCI_LogicAnalyzer 12

9 PCI design (plug-and-play) 13
9.1 PCI_PnP 13
9.2 Load the WDM driver 13

10 Ethernet 14
10.1 Ethernet reference design 14
10.2 Ethernet board setup 14
10.3 Ethernet HDL reference design 14
10.4 Ethernet UDP C code 15

11 I2C bus 16
11.1 I2C controller 16
11.2 I2C onboard devices 16
11.3 I2C functions 16
11.4 Control I2C from USB 16

11.5 Example of I2C communication..... 17
12 LCDs.....18
12.1 Text LCD modules connection..... 18
12.2 Graphic LCD modules connection.....18
13 Mechanical drawing.....19

1 Welcome

1.1 This guide

Welcome to the KNJN PCI FPGA Dragon development board guide.

This guide is divided into short, easy-to-read chapters and explains how to work with your new FPGA board.

1.2 USB driver installation

Dragon is controlled over USB, so let's install the USB driver now. The EzUSB and CyUSB drivers are provided to support all versions of Windows (Me / 98 / 2000 / XP / Vista / 7 / 8 / 8.1). For 64-bit Windows, the driver is not signed so driver signature enforcement must be disabled. For example, see this page for Windows 8.1.

- Connect the Dragon board to the PC USB port. Windows detects the board automatically and starts the "Hardware wizard".
- Instruct the wizard to use the driver files. If Windows asks about the driver not being Windows-certified, select "continue anyway".
- With Windows 7 and above, the wizard fails. The workaround is to install the driver from the Device Manager.

1.3 Power connection

The board is powered from the USB cable by default, but it can also be powered from PCI.

- When USB is used, it provides power to the Dragon board automatically.
- When PCI is used, it does not power the board by default, but a jumper can be installed for that purpose (see section 7.3).

1.4 Software tools

Many files included with the Dragon board are in source-code form.

You need the following tools to compile them:

1. Xilinx ISE WebPack 10.1 + service pack 3
2. Microsoft Visual C++ 5.0 or 6.0, or any other Win32 compiler like lcc-win32 or Digital Mars
3. Microsoft DDK or WDK (optional, if you want to recompile Dragon's PnP driver).

1.5 Linux support

Although Linux is not officially supported by KNJN, the Dragon board can be used on Linux thanks to third-party software support. In particular, Pierre Ficheux's examples can be found here.

2 FPGA configuration

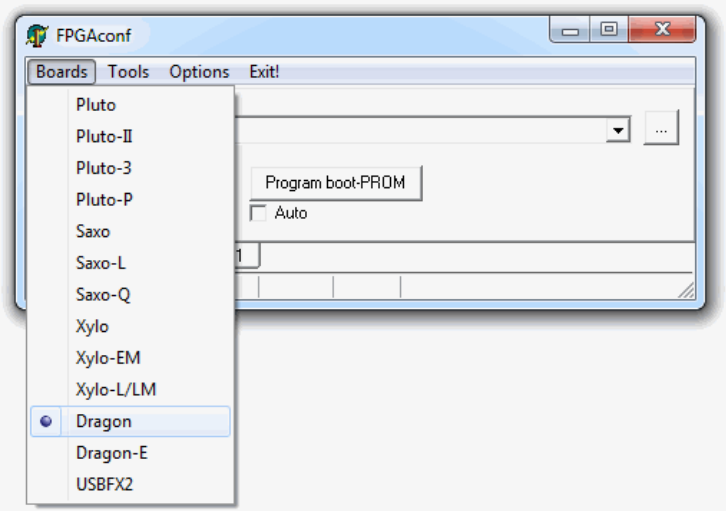
You configure the Dragon board using Xilinx ".bit" files (called BIT files in this document).
BIT files are generated by Xilinx's ISE/WebPack software.

2.1 FPGA configuration using USB

Sample BIT files "ledblink" and "ledglow" are provided in the "Sample files – FPGA" folder.

For a first try, let's use one of them.

1. Run FPGAconf.
2. Select **Boards → Dragon**



3. If you use the CyUSB driver, select these two options:
   - **Options → USB driver → CyUSB**
   - **Options → CyUSB GUID → Cypress**
4. Choose either the ledblink or ledglow BIT file and click the **Configure!** button. The configuration process takes less than one second. On success, both LEDs should blink/glow.

2.2 FPGA configuration using JTAG

The FPGA can also be configured through JTAG (using the Xilinx ISE iMPACT software), although this is slower than USB and requires a separate JTAG cable.

2.3 Boot-PROM

Dragon has a boot-PROM soldered on the backside of the board. At power-up, the FPGA is configured automatically from the boot-PROM. If the PROM is empty or its content is invalid, the FPGA doesn't get configured.

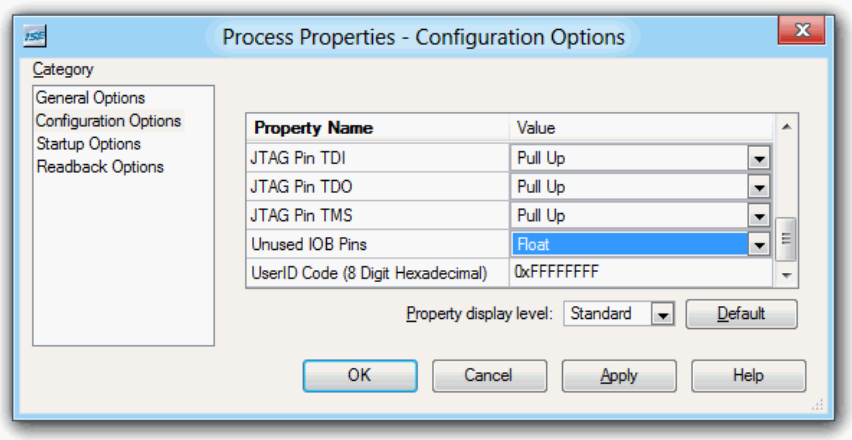
The PROM comes pre-loaded from the factory with a simple version of ledglow.

To program the boot-PROM, run FPGAconf, select the BIT file of your choice and either program, verify or erase the boot-PROM (right-click on the boot-PROM button).

3 FPGA projects with Xilinx's ISE

To start an FPGA project with ISE, proceed as follows:

1. Create a new project.
   In the New Project Wizard, select the "Spartan2" family and the "XC2S100" device in a "TQ144" package. Optionally select some source files before finishing the wizard.
2. Select the top-level design in your project, right-click on "Generate Programming File" and choose "Properties". In "Configuration Options", set "Unused IOB Pins" to "Float". This prevents the FPGA from driving the pins that are not used in your project. Otherwise, grounding all the unused pins (the default) often ends up creating IO contention. You may change this option back once you know that all the pins of your project are correctly assigned.



4 Pins and headers

4.1 FPGA main pins

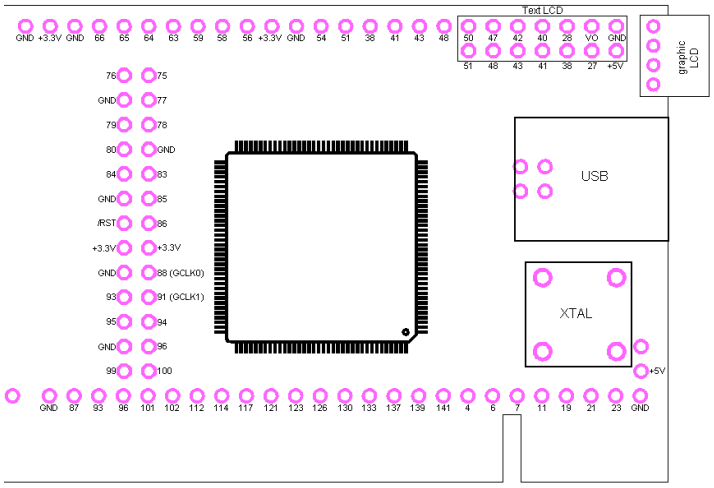
Dragon's important FPGA pins are:

Pin name	Pin number	Comment
CLK_USB	18	24MHz
CLK_XTAL	15	Comes from the optional socketed DIL8 oscillator "XTAL" located below the USB connector
LED1	27	LED on the right of the board (below the USB connector)
LED2	51	LED on the top of the board (above the FPGA)
LED3	75	LED on the left of the board (above the Ethernet connector)

CLK_USB is generated by the USB controller and can be used as a general-purpose clock (24MHz), even if USB is not used.

4.2 FPGA IOs

Note that a few FPGA pins are dual-purpose (they go to two headers).



5 USB

5.1 USB controller

Dragon has a USB controller (AN2135SC) on the back of the board.

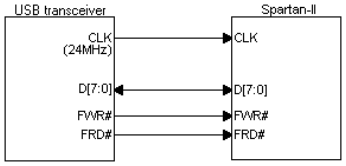
The USB is convenient to:

- Configure the FPGA from a PC (section 2.1)
- After configuration, for PC ↔ FPGA communication (next section).

5.2 USB data interface

After configuration, the USB data interface can be used by the PC to communicate with the FPGA.

USB is natively a serial interface, but Dragon's USB controller serializes/de-serializes the USB data.



The controller uses an 8-bit synchronous data bus with the FPGA.

5.3 USB protocol

The USB protocol is very simple.

The controller provides FWR# and FRD# (active low) signals:

- When the USB controller is idle, FWR# and FRD# are de-asserted (=high).
- When the USB controller sends one byte to the FPGA, it asserts FWR# (=low).
- When the USB controller reads one byte from the FPGA, it asserts FRD# (=low).
- FWR# and FRD# are never asserted (=low) together.

See the next chapter for examples.

6 USB examples

6.1 USB IO8 example

This example reads 8 pins and writes 8 pins from USB. See the “USB\IO8\USB_IO8.v” file in Dragon’s startup-kit.

To output the data to Dragon’s LEDs, just add something like

```
assign LED1 = USBdata[0];
assign LED2 = USBdata[1];
assign LED3 = USBdata[2];
```

with the proper pin assignments (LEDs are pins 27, 51 and 75).

6.2 USB Text LCD example

This example controls a text LCD. See the “USB\LCD_Text_HDL\LCD_Text.v” file in Dragon’s startup-kit.

The code is similar to the one published at http://www.fpga4fun.com/LCDmodule.html

Note that Dragon’s LED pins 27 and 51 are shared with some LCD module pins, so you cannot use them together.

See the “USB\LCD_Text_C\LCD_Text.c” file in Dragon’s startup-kit for the complete code.

The code uses the following main user functions:

```
void USB_Open()
void USB_Close()
void USB_Read(void* buffer, WORD buffersize)
void USB_Write(void* buffer, WORD buffersize)
```

The LCD example adds two helper functions:

```
void USB_WriteChar(char c)
void USB_WriteWord(WORD w)
```

6.3 USB Register banks example

See the “USB\USB_reg_banks” directory in Dragon’s startup-kit for the complete code.
This example shows how to create banks of registers in the FPGA that can be written from USB.

7 PCI

7.1 PCI

PCI allows fast, low-latency communication with a PC. Dragon includes a PCI connector that is compatible with 3.3V and 5V PCI buses (thanks to the two slots in the PCI connector). Note that the **Dragon USB connector should face the inside of the computer** (and the Dragon Ethernet connectors should face the outside).

PCI doesn't allow plugging a board into a running system. Before plugging Dragon into (or unplugging it from) a computer's PCI bus, make sure the computer is off.

7.2 PCI and USB

While doing PCI development with Dragon, it is convenient to also use the USB connection (for FPGA configuration).

USB and PCI may be connected to 2 different PCs. In this case, try to get the ground potentials between them as equal as possible (for example by powering the two PCs from the same power strip).

Note that the FPGA can be configured in a running system, as the FPGA pins go into a high-impedance state during configuration (so even if the same PC is used for PCI and USB, the PC can still reconfigure its own FPGA).

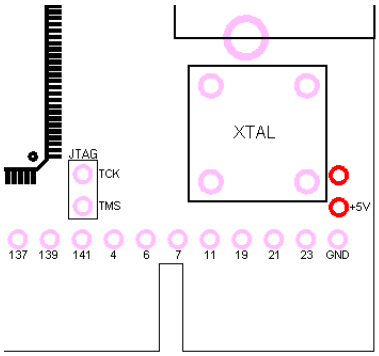
7.3 PCI power

Dragon can be powered from the PCI bus by installing a wire jumper. This lets you use Dragon as a standalone PCI board, for example (without a USB connection).

Note that USB alone is usually enough to power Dragon, so the PCI power option is not required when USB is used.

Enabling the PCI power option:

Here is a view of the lower-right corner of the board. To power Dragon from PCI, solder a wire or jumper across the two red pads.



If USB and PCI are used with different PCs, and the PCI powered option is enabled, the +5V of both PCs are connected together. To prevent one system from powering the other through the USB cable, use the following strategy: make the USB connection only after all systems have powered-up, and break the USB connection before any system powers-down.

7.4 PCI BIOS

By default, newer PC BIOSes disable the CLK signal of PCI slots where they believe no card is present. A BIOS may not detect Dragon if you use a PCI core without configuration space.

If your BIOS has an "Auto Detect DIMM/PCI Clk" option, disable it.

Some BIOSes don't allow this option to be disabled. The PCI_PnP design (9.1) implements configuration space, so it should work in all cases.

8 PCI designs (non plug-and-play)

8.1 PCI designs

The designs are introduced in order of increasing complexity.

The designs are provided in Verilog HDL source form (Verilog + UCF file). You have to create ISE projects to compile them with Xilinx ISE/WebPack. Use ISE's New Project Wizard and don't forget to include the UCF file in each ISE project, as the UCF files contain important pin assignments and timing constraints.

An example of how to create an ISE project is given at http://www.fpga4fun.com/ISEQuickStart.html, but instead of creating a new UCF file, use the ones provided.

All these PCI designs can be tried using 1 or 2 computers (see also section 7.2 when using 2 computers).

8.2 PCI_IORAM

This design implements a small 32-bit x 16-location RAM in the FPGA, in IO space.

The design uses 16 32-bit addresses, from 0x0200 to 0x023F by default. The start address is defined by this line in the Verilog file:

```
parameter IO_address = 32'h00000200;
```

Change this address if your PCI PC already has peripherals at these addresses.

See fpga4fun's site for details and related software at http://www.fpga4fun.com/PCI2.html. In particular, UserPort or GiveIO may be required (provided in the startup-kit).

8.3 PCI_IORAM_LCD

This design adds a simple text LCD module control logic.
Use C code similar to:

```
WriteIO_DWORD(0x200, 0x38);   sleep(5);
WriteIO_DWORD(0x200, 0x0F);   sleep(5);
WriteIO_DWORD(0x200, 0x01);   sleep(100);

WriteIO_DWORD(0x204, 'H');   sleep(1);
WriteIO_DWORD(0x204, 'e');   sleep(1);
WriteIO_DWORD(0x204, 'l');   sleep(1);
WriteIO_DWORD(0x204, 'l');   sleep(1);
WriteIO_DWORD(0x204, 'o');   sleep(1);
WriteIO_DWORD(0x204, ' ');   sleep(1);
WriteIO_DWORD(0x204, 'P');   sleep(1);
WriteIO_DWORD(0x204, 'C');   sleep(1);
WriteIO_DWORD(0x204, 'I');   sleep(1);
WriteIO_DWORD(0x204, ' ');   sleep(1);
WriteIO_DWORD(0x204, '!');   sleep(1);
WriteIO_DWORD(0x204, '!');   sleep(1);
```

The design sends LCD commands using IO Writes at address 0x200, and LCD data using IO Writes at address 0x204.

```
parameter IO_address = 32'h00000200;
```

Change this address if the PCI PC already has peripherals at these addresses.

8.4 PCI_LogicAnalyzer

This design adds extra logic to capture the PCI bus signals. The signals are stored in a 48-bit x 256-deep internal FPGA RAM and then read back through the USB port.

The C source code required to read the data from USB is provided. See also http://www.fpga4fun.com/PCI3.html to download the “PCI bus Viewer” application.

9 PCI design (plug-and-play)

9.1 PCI_PnP

This design adds registers into PCI configuration space and works together with the provided PCI WDM driver. This allows Dragon to be detected by the Windows device manager, and memory resources to be allocated by Windows.

9.2 Load the WDM driver

"PCPCI" is the PC connected to Dragon's PCI bus, and "PCUSB" the PC connected to Dragon's USB port (PCPCI and PCUSB may be the same computer).

Follow these instructions.

1. Connect Dragon's USB port to PCUSB.
2. Turn off PCPCI, plug Dragon into one of its free PCI slots and power on PCPCI.
3. With PCUSB, open the "PCI_PnP.v" file using a text editor.
   Check the "VENDOR_ID" and "DEVICE_ID" values. They look like this:
   ```
   parameter VENDOR_ID = 16'h0100;
   parameter DEVICE_ID = 16'h0000;
   ```
4. With PCPCI, open the "DragnPCI.inf" file using a text editor, and make sure the VENDOR_ID and DEVICE_ID values are the same. They look like this:
   ```
   PCI\VEN_0100&DEV_0000&SUBSYS_00000000&REV_00
   ```
5. With PCUSB, compile the "PCI_PnP" design (using Xilinx ISE) and configure the Dragon board (through USB) while it is plugged into PCPCI.
6. With PCPCI, open the Windows Device Manager. A tree of devices is shown.
   Right-click on the computer's icon and select "Scan for hardware changes" (or "Refresh" for Win98)
7. PCPCI detects new hardware and the "New Hardware Wizard" starts.
   WinXP's wizard calls the new device "PCI device" while Win98's wizard calls it an "Early non-VGA Device" (unless you changed the Device "Class Code" inside the "PCI_PnP.v" file).
   The wizard differs slightly between Windows versions; continue it until you can specify the driver files location (DragnPCI.inf and DragnPCI.sys).
   Select "Continue anyway" if asked.
8. PCPCI installs the driver and creates a new device entry. Go to the device's properties and check the resources tab. Windows should have allocated some memory or IO resources.
9. Run the "DragonPCI.exe" application on PCPCI.
   It is a very simple example of reads and writes using the WDM driver.

All C source code is provided (DragonPCI WDM driver and DragonPCI application).

10 Ethernet

Dragon can accommodate up to two RJ-45 connectors.

10.1 Ethernet reference design

A reference design is provided that uses one RJ-45 connector and allows bi-directional UDP 10BASE-T communication. The second RJ-45 could be used to implement advanced functions like a network router or network filter/firewall.

10.2 Ethernet board setup

The design uses the top RJ-45 connector. If your board doesn't have one fitted already, solder an RJ-45 connector on the board. A connector with integrated magnetics is recommended, but in a lab environment the design works fine with a regular connector.

Connect a network cable from a network hub or switch to the Dragon's RJ-45. If you want to connect Dragon directly to a PC network card, use a "crossover" network cable.

The reference design uses a 40MHz oscillator. Plug it into the "XTAL" socket on the Dragon board (pay attention to the oscillator orientation).

10.3 Ethernet HDL reference design

The reference design is provided in source code form only.

The design provides an example of UDP/IP transmission and reception.

- Transmission: a packet is sent at regular intervals (about every 2 seconds) with a fixed 18-byte payload.
- Reception: every time a UDP packet is received, Dragon checks the packet validity and updates its LEDs (the first 3 bits of the UDP payload are used to update the 3 LEDs).

Before synthesizing the HDL, make sure you update the IP and PA parameters inside the source code. In particular:

- IP: "IP destination" - put the IP of the PC you want to send to.
- PA: "Physical Address" - put the address of the PC you want to send to.
- myIP - IP of Dragon. Choose an unused IP that is compatible with your network.[1]

If the PC has trouble receiving from Dragon:

- Make sure the hub/switch light blinks every 2 seconds or so, to indicate that a packet is transmitted by Dragon
- Make sure you are not running a firewall on the PC

If the PC has trouble sending to Dragon:

Make sure the PC has the correct ARP entries. You can use "ARP –a" to check the ARP entries. The physical address of Dragon should be listed. If it is not, you can add it manually using "ARP" or "netsh". See the examples below for an FPGA at IP 192.168.0.44 with physical address 00-12-34-56-78-90:

- For Windows XP:

  ARP –s 192.168.0.44 00-12-34-56-78-90

  Note that an additional IP parameter can be used in the ARP -s command to specify the IP of the interface where your FPGA board is plugged. This may be required if your machine has multiple Ethernet ports that are enabled concurrently.

[1] If in doubt, choose an IP whose first 3 numbers match the other devices on your network, and set the fourth number to a random value between 1 and 254. Then "Ping" the new IP to make sure it is not in use.

- For Windows Vista / 7 / 8 / 8.1:
  netsh interface ipv4 add neighbors "Local Area Connection" 192.168.0.44 00-12-34-56-78-90

  Make sure you open an administrator command-line prompt (otherwise the netsh command fails).

10.4 Ethernet UDP C code

Here is a simple UDP transmit example:

```
#include <winsock.h>
#include <stdio.h>
#pragma comment (lib, "wsock32.lib")

char* szUDPAddress = "192.168.0.44";
u_short UDPPort = 1024;
char szMessage[1500] = "Whoa!";

int main(int argc, int **argv)
{
 WSADATA wsda;
 int ret;

 SOCKET s;
 SOCKADDR_IN addr;

 WSAStartup(MAKEWORD(1,1), &wsda);
 s = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
 if(s==SOCKET_ERROR) { printf("Failed to create socket, error %d\n", WSAGetLastError()); exit(1); }

 addr.sin_family = AF_INET;
 addr.sin_port = htons(UDPPort);
 addr.sin_addr.s_addr = inet_addr(szUDPAddress);

 ret = sendto(s, szMessage, strlen(szMessage), 0, (struct sockaddr *) &addr, sizeof(addr));
 if(ret == SOCKET_ERROR) { printf("Failed to send, error %d\n", WSAGetLastError()); exit(1); }

 closesocket(s);
 WSACleanup();
 return 0;
}
```

See more examples in the Ethernet\UDP directories.

11 I2C bus

Dragon has an integrated I2C bus controller.

11.1 I2C controller

The I2C bus controller is a “hard macro” (the hardware is built into the USB controller), as opposed to a “soft macro” I2C controller that can be built into the FPGA.

Dragon also has an I2C EEPROM on board.

11.2 I2C onboard devices

The following devices share the I2C bus:

1. I2C controller (AN2135SC)
2. 64Kbits (8Kbytes) EEPROM (Microchip 24AA64)
3. FPGA pins 65 (SCL) and 66 (SDA)
4. Two connectors on the top left of the board, to connect to external devices.

The FPGA can implement a “soft macro” I2C controller and communicate with the I2C bus. The two connectors allow extending the bus externally.

11.3 I2C functions

The “hard macro” I2C controller can issue multiple commands:

1. Write packets
2. Read packets
3. Multiple Read/Write packets
4. I2C bus scan

11.4 Control I2C from USB

The I2C controller is under USB control, so it is easy to control from a PC.

1. To send a USB I2C command, use USB pipe 4.
2. Immediately after, read the response from USB pipe 3.

Both command and response are limited to 64 bytes.

USB I2C command format:
First byte: number of blocks
Then for each block:

1. If I2C read: Number of bytes to read, plus I2C address (with lsb=1 to indicate a read)
   On return, you get one status byte plus the data read.
2. If I2C write: Number of bytes to write (including the I2C address), followed by the I2C address (with lsb=0), plus the bytes to write.
   On return, you get one status byte.

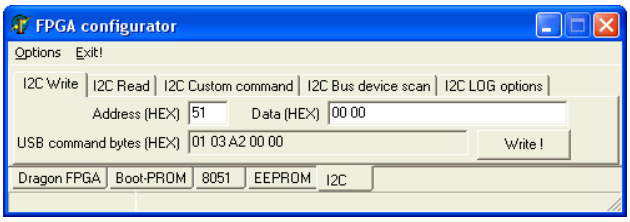
Status byte:
bit 7: BERR (bus error), indicates a bus contention (another I2C master took control of the bus)
bit 6: ACK, set if the device is present, cleared if the device didn’t respond, or asked to stop
bit 5-0: byte count, number of bytes written or read

If a block returns an error (BERR or no ACK), the I2C controller stops and doesn’t process any further blocks in the packet.

11.5 Example of I2C communication

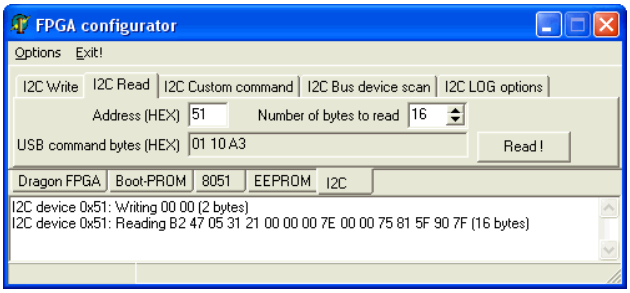
The FPGAconf application makes it easy to exercise the I2C bus from the PC.

Let's communicate with the onboard I2C EEPROM. The EEPROM is at I2C address 0x51. First write "00 00" to the EEPROM. This resets its internal address counter (don't enter more than two "00" as this would start writing to the EEPROM content).



Now go to the "I2C Read" tab, and request a 16-byte read.

You should get the following:



You can also combine multiple I2C reads and writes into a single command. Go to the "I2C Custom command" tab, and enter the following command: 02 03 A2 00 00 10 A3

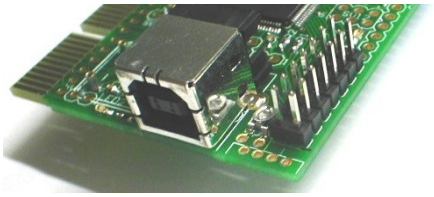
This combines the 2 previous operations. See 11.4 for more information on the protocol.

12 LCDs

12.1 Text LCD modules connection

Common text LCD modules can be connected using the dedicated 2x7=14-pin connector on the upper side of the board, above Dragon's USB connector.

Here is the connector fitted with a male header.



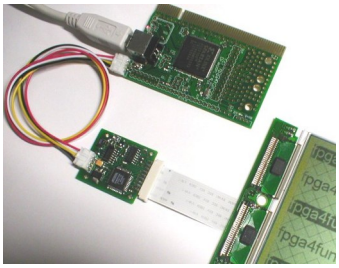
A small potentiometer below the connector allows changing the LCD contrast.

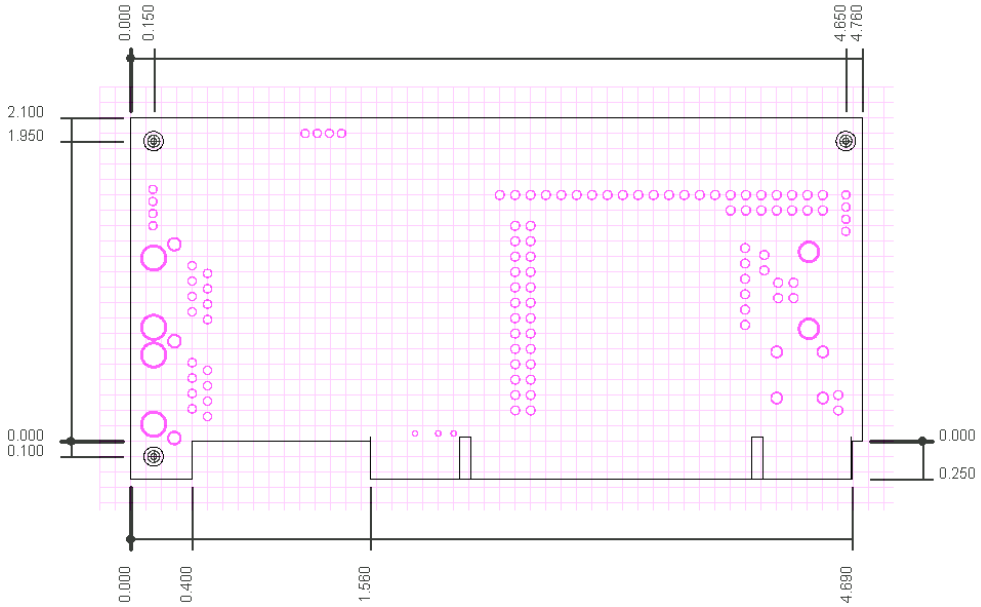
Take care not to connect an LCD module with the pins reversed. See this page for more details.

12.2 Graphic LCD modules connection

Since there is no industry-standard graphic LCD module connector, a custom connection needs to be made.

Dragon has a small 4-pin connector next to the USB connector that can be used to provide video information to specific graphic LCD modules. Get more information here.



13 Mechanical drawing

All dimensions in inches (1 inch = 25.4mm).