

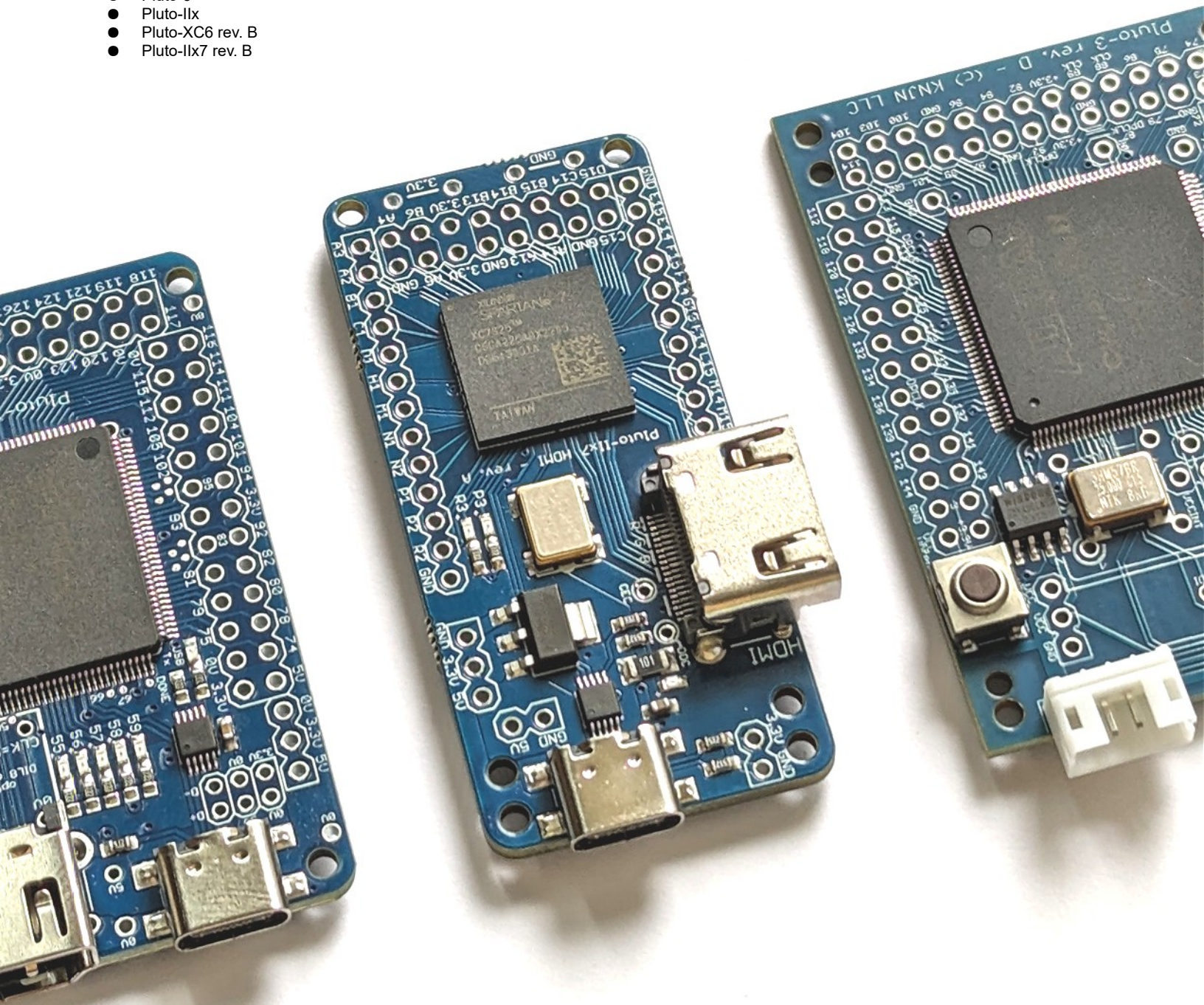
# FPGA serial development boards

© 2004 to 2026 KNJN LLC

<https://www.knjin.com/>

This document applies to the following boards:

- Pluto
- Pluto-3
- Pluto-llx
- Pluto-XC6 rev. B
- Pluto-llx7 rev. B



Document last revision on **March 24, 2026**

---

# Table of Contents

1 Welcome.....	4
1.1 The Pluto family.....	4
1.2 Software.....	4
2 FPGAconf.....	5
2.1 FPGA configuration.....	5
2.2 FPGA boot-PROM.....	5
2.3 Auto configuration mode.....	5
3 FPGA project for Pluto/Pluto-3.....	6
3.1 Create a new project.....	6
3.2 A simple start.....	6
4 FPGA project for Pluto-IIx/Pluto-XC6/Pluto-IIx7.....	7
4.1 Create a new project.....	7
4.2 A simple start.....	7
5 FPGA connections.....	8
5.1 FPGA pins.....	8
5.2 Boot-PROM connection.....	8
5.3 Secondary connector.....	8
5.4 Power header.....	8
5.5 JTAG connection.....	8
6 Clock sources.....	9
6.1 High-stability clock.....	9
6.2 GPSDO.....	9
7 Flashy boards.....	10
7.1 FlashyMini design.....	10
7.2 FlashyDemo design.....	10
8 FPGA configuration.....	11
8.1 Pluto/-IIx/-XC6/-IIx7 FPGA configuration.....	11
8.2 Pluto-3 FPGA configuration.....	11
9 Sample C code for serial Win32 send & receive.....	12
10 Board dimensions and IO pin assignment.....	13
10.1 Pluto.....	13
10.2 Pluto-3.....	14
10.3 Pluto-IIx / HDMI.....	15
10.4 Pluto-XC6 HDMI.....	16
10.5 Pluto-IIx7 HDMI.....	17



# 1 Welcome

## 1.1 The Pluto family

The Pluto boards are controlled through a serial port.

The serial port allows FPGA configuration, and after configuration host/PC ↔ FPGA communication.

	<b>Pluto</b>	<b>Pluto-3</b>	<b>Pluto-IIx</b>	<b>Pluto-XC6</b>	<b>Pluto-IIx7</b>
<b>FPGA</b>	EP1K10	EP2C5 or EP2C8	XC3S50A or XC3S200A	XC6SLX9	XC7S25
<b>FPGA datasheet</b>	<a href="#">ACEX 1K</a>	<a href="#">Cyclone II</a>	<a href="#">Spartan-3A</a>	<a href="#">Spartan-6</a>	<a href="#">Spartan-7</a>
<b>HDMI</b>	no	no	option	yes	yes
<b>Logic cells</b>	576	4608 or 8256	1584 or 4032	9152	23360
<b>IO pins</b>	41	65	48 or 36+HDMI	62 + HDMI	32 + HDMI
<b>LEDs (note 1)</b>	1	2	1 or 2	5	2
<b>Boot-PROM (note 2)</b>	-	4Mbit	4Mbit	8Mbit	16MBit
<b>External clocks</b>	Up to 2	Up to 4	Up to 4	Up to 8	Up to 7
<b>On-board oscillator</b>	25MHz	25MHz	25MHz	25MHz	25MHz
<b>Dimensions</b>	58 x 28 mm	58 x 41 mm	58 x 28 mm	58 x 41 mm	58 x 28 mm

Note 1: Boards with USB-C port have an additional LED that shows the USB TxD activity.

Note 2: Minimum boot-PROM size shown here. Actual product may use a higher capacity boot-PROM.

## 1.2 Software

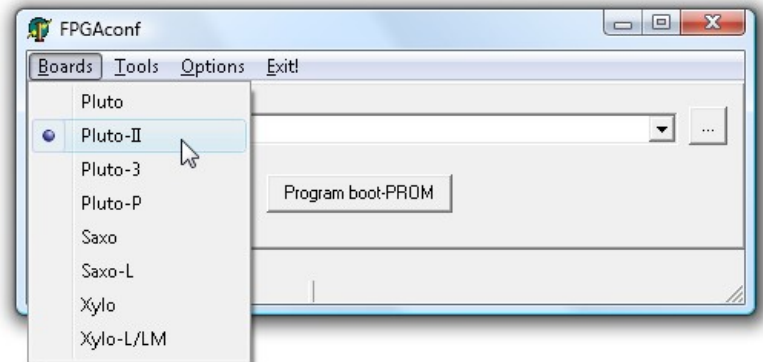
Once your board has shipped, check your order page for links to the FPGA vendor software and the “startup-kit” that includes utilities and example source code.

## 2 FPGAconf

### 2.1 FPGA configuration

FPGAconf is a utility provided in the startup-kit.

1. Select your board in the “Boards” menu.
2. Choose an FPGA bitfile (click on the browse button “...”).
3. Click “Configure FPGA”.

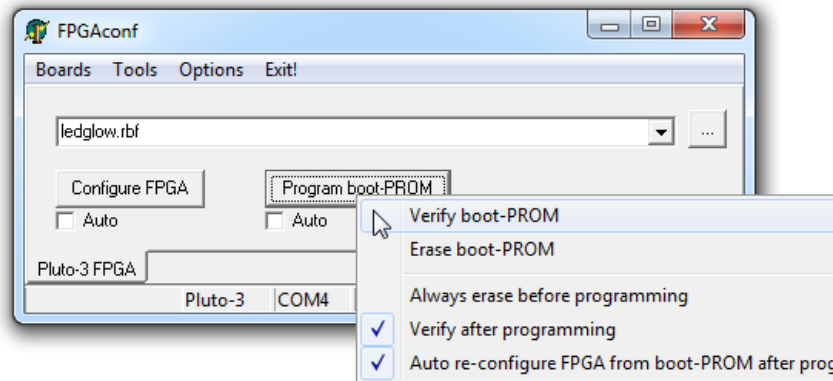


### 2.2 FPGA boot-PROM

The boot-PROM is the memory that configures the FPGA at power-up.

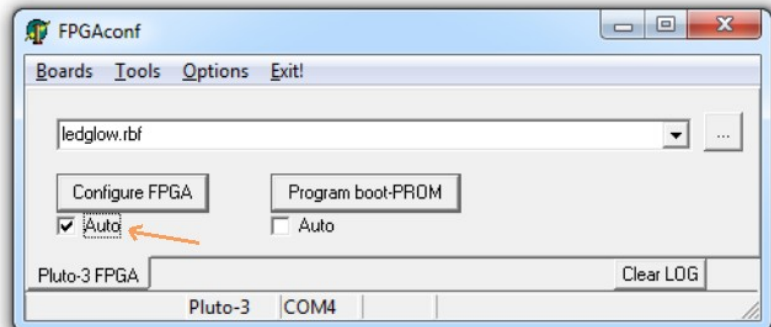
If the boot-PROM is empty or its content is invalid, the FPGA stays un-configured and ready for manual configuration.

The boot-PROM can be programmed, verified and erased (click on the “Program boot-PROM” button for programming, right-click for verifying or erasing).



### 2.3 Auto configuration mode

When the “Auto mode” is enabled, FPGAconf takes action each time the bitfile is updated. Useful for example if you want to re-configure the FPGA automatically after each compilation.



## 3 FPGA project for Pluto/Pluto-3

Pluto and Pluto-3 are configured from SOF or RBF files generated by Altera's Quartus-II software.

### 3.1 Create a new project

1. Run Quartus-II, and click on menu → File → New Project Wizard.
2. Select the project location, choose a project name, and click Next.
3. Choose files to add to the project. Just click next if you don't have files to add now.
4. Now is time to choose the device (you can also do that later using menu → Assignments → Device)
  - a. For Pluto, choose family "APEX1K" and device "EP1K10TC100-3".
  - b. For Pluto-3, choose family "Cyclone-II" and device "EP2C5T144C8".
5. Click Finish.

A graphical work-through is also available on [this fpga4fun page](#).

### 3.2 A simple start

Here's a simple Verilog file:

```
module LEDblink(  
    input clk,  
    output LED  
);  
  
reg [31:0] cnt;  
always @(posedge clk) cnt <= cnt + 1; // 32 bits counter  
  
assign LED = cnt[23];  
endmodule
```

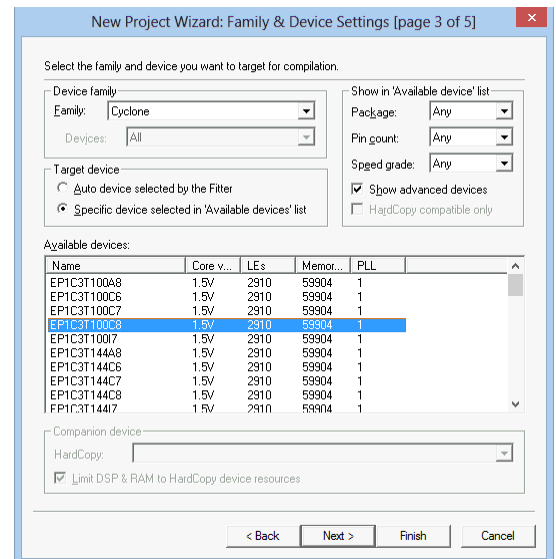
Add it to the project and select it as the top-level design. Next make the correct pin assignments in Quartus-II menu → Assignments/Pins or "Pin planner" (using the info from paragraph 5.1). This project uses only 2 pins, so it should be fast.

You also want to specify the outputs and what happens to unused pins.

1. Select menu → Assignments → Device
2. Click on "Device & Pin Options..."
  - a. Go to the "Programming Files" tab, select "Raw Binary File (.rbf)".
  - b. Go to Unused Pins", select "As inputs, tri-stated" or "As inputs with weak pull-up".
  - c. Click "OK".
3. Click "OK".

Option 2.a makes sure RBF files are generated (used with FPGAconf). Otherwise only SOF files are generated (used for JTAG).

Option 2.b prevents the FPGA from driving pins that are not used in your project. Otherwise, Quartus-II drives all the unused pins to ground, which may create IO contentions.



## 4 FPGA project for Pluto-IIx/Pluto-XC6/Pluto-IIx7

The Pluto-IIx, Pluto-XC6 and Pluto-IIx7 boards are configured from “.bit” files generated by Xilinx software.

### 4.1 Create a new project

Run ISE or Vivado and create a new project.

1. For Pluto-IIx, select the “Spartan3A and Spartan3AN” family and the device on your board (XC3S50A or XC3S200A) in VQ100 package.
2. For Pluto-XC6, select the Spartan-6 family, then the XC6SLX9 device in TQG144 package.
3. For Pluto-IIx7, select the XC7S25 in CSGA225 package.

You can now create or add source files in the project.

### 4.2 A simple start

Here’s a simple Verilog file:

```
module LEDblink(  
    input clk,  
    output LED  
);  
  
    reg [31:0] cnt;  
    always @(posedge clk) cnt <= cnt + 1;    // 32 bits counter  
  
    assign LED = cnt[23];  
endmodule
```

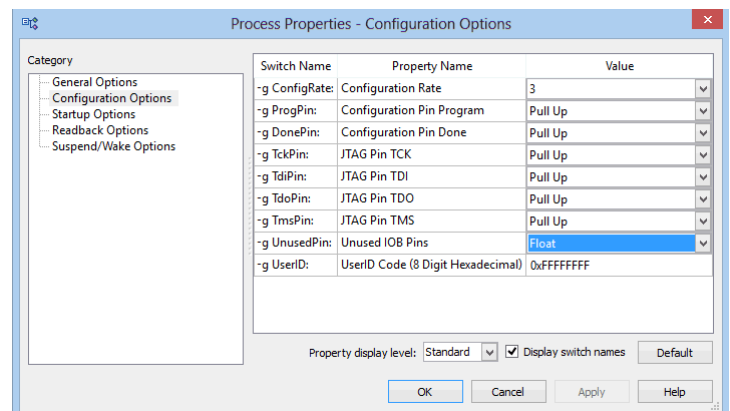
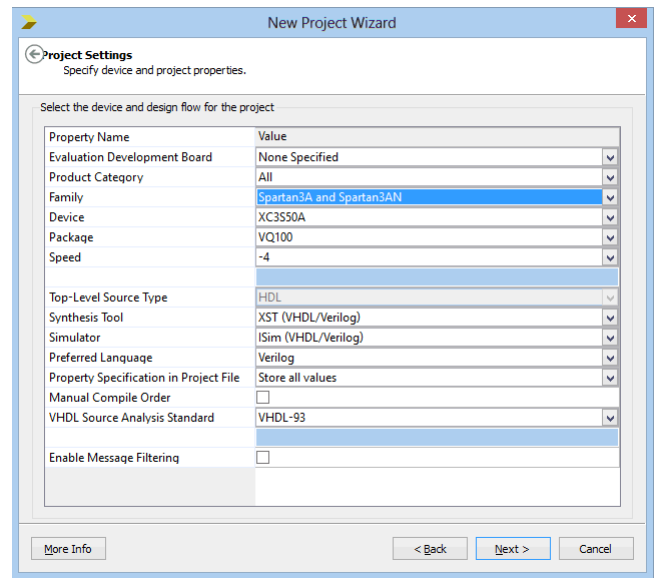
Add it to the project and select it as the top-level design in your project.

With ISE, add a UCF file to the project. For example, use this for a Pluto-IIx

```
NET "clk" LOC = P40;  
NET "LED" LOC = P29;
```

and right-click on “Generate Programming File”, choose “Process Properties”, then click on “Enable BitStream Compression” and allow “Unused IOB Pins” to “Float” or “Pull-up”.

With Vivado, add an XDC file to the project (examples are provided in the startup-kit).



## 5 FPGA connections

### 5.1 FPGA pins

The main FPGA signals are:

Pin name	Pluto	Pluto-3	Pluto-Ilx	Pluto-XC6	Pluto-Ilx7	Direction	Comment
CLK0	91	17	40	127	M9	FPGA input	25MHz on-board SMD oscillator
CLK1		18		51		FPGA input	Optional DIL8 oscillator
CLK2				88		FPGA input	Optional SMD 3x2 oscillator
LED1	7	28	29	59	R3	FPGA output	LED (active high)
LED2		25		58	P3	FPGA output	LED (active high)
RxD	77	21	21	61	R5	FPGA input	FPGA receives from PC
TxD	78	24	30	62	R4	FPGA output	FPGA transmits to PC
PB		9				FPGA input	Push-button (active low)

The RxD and TxD pins are used as an asynchronous serial port and allow communication with the PC. Many other IO signals are available on unpopulated headers, see the chapter 10. For HDMI, check the demo project in the startup-kit.

### 5.2 Boot-PROM connection

The boot-PROM is a [W25Q](#) or similar SPI flash.

SPI flash pin	Pluto-3 FPGA pin	Pluto-Ilx FPGA pin	Pluto-XC6 FPGA pin	Pluto-Ilx7 FPGA pin
Clock	15	53	70	F5
Data In	1	46	64	H14
Data Out	14	51	65	H15
nCS	2	27	38	L11
nHOLD	8	31	69	J1

### 5.3 Secondary connector

This 4-pins connector is located on the side of the board. It can be easily soldered (available as KNJN [item#1804](#) or from [DigiKey](#)). It has 2 power pins and 2 IOs.

Pin	Pluto	Pluto-3	Pluto-Ilx	Pluto-XC6	Pluto-Ilx7	Comment
1	VCC-unreg (1)	VCC-unreg (1)	3.3V	5V	N/A	Power
2	FPGA IO pin 8	FPGA IO pin 30	FPGA IO pin 43	FPGA IO pin 34	N/A	RxD or serclk or other use
3	FPGA IO pin 96	FPGA IO pin 31	FPGA IO pin 44	FPGA IO pin 35	N/A	TxD or serdata or other use
4	GND	GND	GND	GND	N/A	Ground

(1) VCC-unreg is +5V when using USB, otherwise it is the voltage used to power your FPGA board with, typically +5V to +10V.

### 5.4 Power header

This 3-pins header provides access to the board power signals. It is often used as an output (to power other boards) but can also be used as an input (to power the Pluto board).

### 5.5 JTAG connection

Board	JTAG
Pluto	JTAG is not available
Pluto-3	Pluto-3 has a full-size Altera-style 10 pins header. A matching shrouded connector is easily soldered (like KNJN items <a href="#">2450</a> or <a href="#">2451</a> )
Pluto-Ilx	The JTAG signals are accessible on two 2x3 headers next to the FPGA
Pluto-XC6	The JTAG signals are accessible on one 1x6 header below the FPGA (bottom side of the board)
Pluto-Ilx7	JTAG is not available

---

## 6 Clock sources

The boards main clock is a 25MHz SMD oscillator.

### 6.1 High-stability clock

If a high-stability clock is desired with Pluto-XC6 and Pluto-Ilx7, the board can be fitted with a VCTCXO. These oscillators are quite stable, typically  $\pm 2.5$ ppm or better.

- With Pluto-XC6, solder a VCTCXO on the dedicated SMD 3x2 oscillator footprint located on the back of the Pluto-XC6 board (marked P88). Or order “Pluto-XC6 HSC” for the VCTCXO pre-assembled.
- With Pluto-Ilx7, replace the main 25MHz oscillator with a compatible VCTCXO. Or order “Pluto-Ilx7 HSC” for the VCTCXO pre-assembled.

### 6.2 GPSDO

For further accuracy, a GPS module can be used to control the VCTCXO and create a GPSDO. Make sure to use a five pins GPS module (with PPS output). Then run the GPSDO example code.

Boards with assembled VCTCXO and GPS modules are available on the [FPGA kits](#) page.

## 7 Flashy boards

With an optional Flashy ADC board, the system becomes a digital oscilloscope.

### 7.1 FlashyMini design

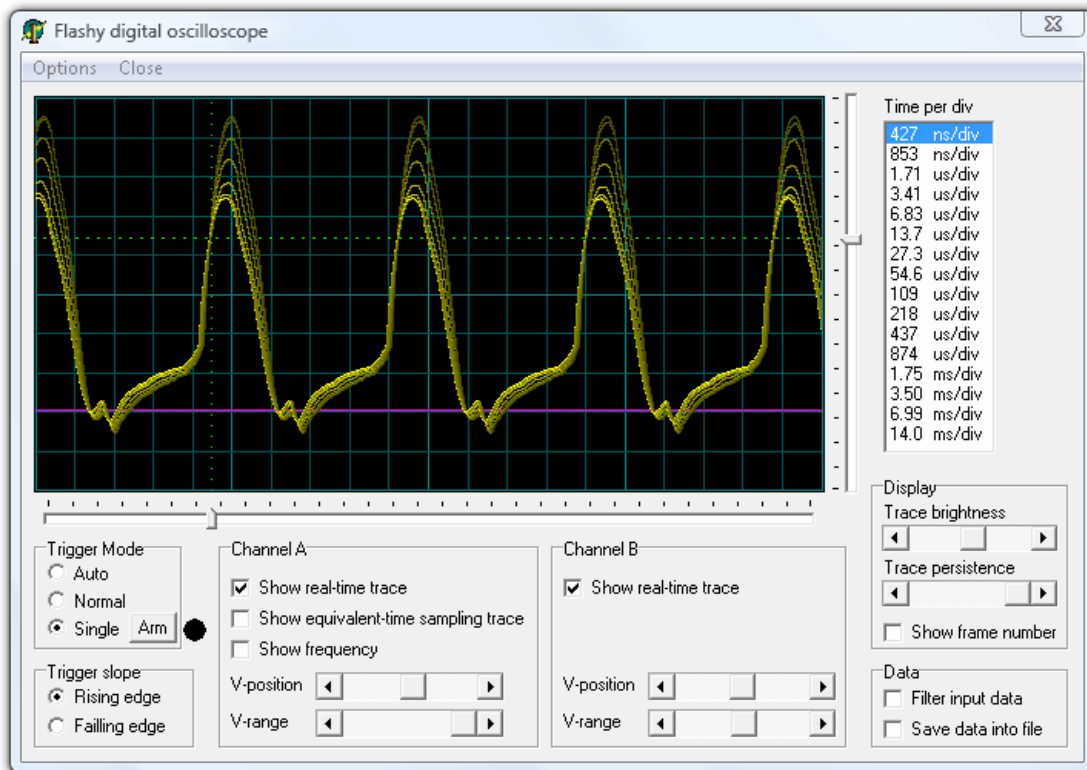
FlashyMini is provided with full source code (HDL + C). It shows how to get data from Flashy and can be used as a skeleton to develop your own acquisition system.

### 7.2 FlashyDemo design

FlashyDemo is provided in binary form. It is a showcase of Flashy possibilities, implementing features found in digital oscilloscopes like pre-trigger acquisition and equivalent-time-sampling.

To run FlashyDemo:

1. Mount Flashy on the Pluto board, and power it up.
2. Configure the FPGA with the FlashyDemo bitfile.
3. Go to Menu → Tools → Flashy Oscilloscope (or press CTRL-F).



Note that there are actually four kind of Flashy boards available (Flash, Flashy, FlashD and FlashyD). Here's the compatibility table.

	Flash	Flashy	FlashD	FlashyD	LCD (2)
<b>Pluto</b>	Limited (1)	Limited (1)	No	No	No
<b>Pluto-3</b>	Yes	Yes	Yes	Yes	Yes
<b>Pluto-IIx</b>	Yes	Yes	No	No	Yes
<b>Pluto-XC6</b>	Yes	Yes	Yes	Yes	Yes
<b>Pluto-IIx7</b>	Yes	Yes	No	No	No

(1) Pluto's FPGA cannot hold all the FlashyDemo functionality at once, so two FlashyDemo bitfiles are provided. Each covers a different set of features.

(2) The KNJN color LCD [item#5300](#) option can work as a FlashyDemo external display.

For more information, check the [Flashy acquisition board](#) page.

---

## 8 FPGA configuration

In case you don't want to use FPGAconf.

### 8.1 Pluto-/llx/-XC6-/llx7 FPGA configuration

Set the baud speed at 115200 bauds for TXDI and 3000000 bauds for USB-C, and run the following C pseudo-code:

For each byte of the bitfile, do:

```
    for(j=0; j<8; j++)    serial.write(~(bufbyte >> j & 1 ^ 1)); // for Altera (LSB first)
or
    for(j=0; j<8; j++)    serial.write(~(bufbyte >> (j^7) & 1)); // for Xilinx (MSB first)
```

A complete example for Altera could be:

```
FILE *fpIn = fopen("LEDblink.rbf", "rb");
char buf[0x100000];
int len = fread(buf, 1, sizeof(buf), fpIn);
fclose(fpIn);

OpenCom("COM3", 115200);
SetCommBreak(hCom); Sleep(50); ClearCommBreak(hCom); // un-configure FPGA

for(int i=0; i<len; i++)
    for(int j=0; j<8; j++)
        WriteComChar(~(buf[i] >> j & 1 ^ 1));

CloseCom();
```

See also the chapter 9 for some extra source code.

### 8.2 Pluto-3 FPGA configuration

Pluto-3's configuration scheme is even simpler. To configure the FPGA, just send the RBF binary content through the serial port at 115200 bauds in 8-bits mode. To un-configure the FPGA (before sending the RBF), send a "break" condition (a high signal) for about 50ms.

On Linux, a script can be used:

```
#!/bin/bash
#
# pluto3configure :: send an rbf file to a Pluto-3 FPGA board
#

SERIAL=/dev/ttyUSB0

if [ ! -f "$1" ]
then
    echo "Usage: $(basename $0) filename.rbf" >&2
    exit 1
fi

(stty 115200 raw cs8 -cstopb -parenb -ixon -crtscts 0<&1 ; sendbreak; dd "if=$1" bs=1k) > $SERIAL
```

## 9 Sample C code for serial Win32 send & receive

```
#include <windows.h>
HANDLE hCom;

void ExitOnError(char*message)
{
    printf("%s error", message);
    exit(1);
}

void OpenCom(char* COM_name)
{
    DCB dcb;
    COMMTIMEOUTS ct;

    hCom = CreateFile(COM_name, GENERIC_READ | GENERIC_WRITE, 0, NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
    if(hCom==INVALID_HANDLE_VALUE) ExitOnError(COM_name); // can't open COM port
    if(!SetupComm(hCom, 4096, 4096)) ExitOnError("SetupComm");

    if(!GetCommState(hCom, &dcb)) ExitOnError("GetCommState");
    dcb.BaudRate = 115200;
    ((DWORD*)&dcb)[2] = 0x1001; // set port properties for TXDI + no flow-control
    dcb.ByteSize = 8;
    dcb.Parity = NOPARITY;
    dcb.StopBits = 2;
    if(!SetCommState(hCom, &dcb)) ExitOnError("SetCommState");

    // set the timeouts to 0
    ct.ReadIntervalTimeout = MAXDWORD;
    ct.ReadTotalTimeoutMultiplier = 0;
    ct.ReadTotalTimeoutConstant = 0;
    ct.WriteTotalTimeoutMultiplier = 0;
    ct.WriteTotalTimeoutConstant = 0;
    if(!SetCommTimeouts(hCom, &ct)) ExitOnError("SetCommTimeouts");
}

void CloseCom()
{
    CloseHandle(hCom);
}

DWORD WriteCom(char* buf, int len)
{
    DWORD nSend;
    if(!WriteFile(hCom, buf, len, &nSend, NULL)) exit(1);

    return nSend;
}

void WriteComChar(char b)
{
    WriteCom(&b, 1);
}

int ReadCom(char *buf, int len)
{
    DWORD nRec;
    if(!ReadFile(hCom, buf, len, &nRec, NULL)) exit(1);

    return (int)nRec;
}

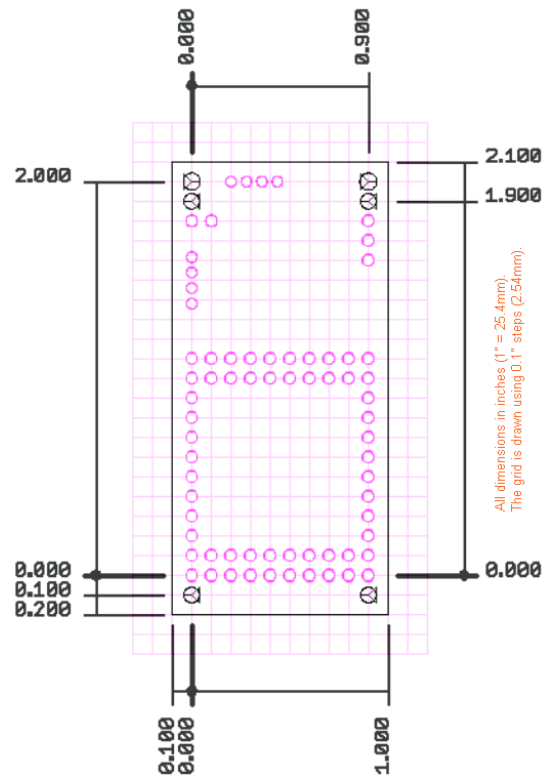
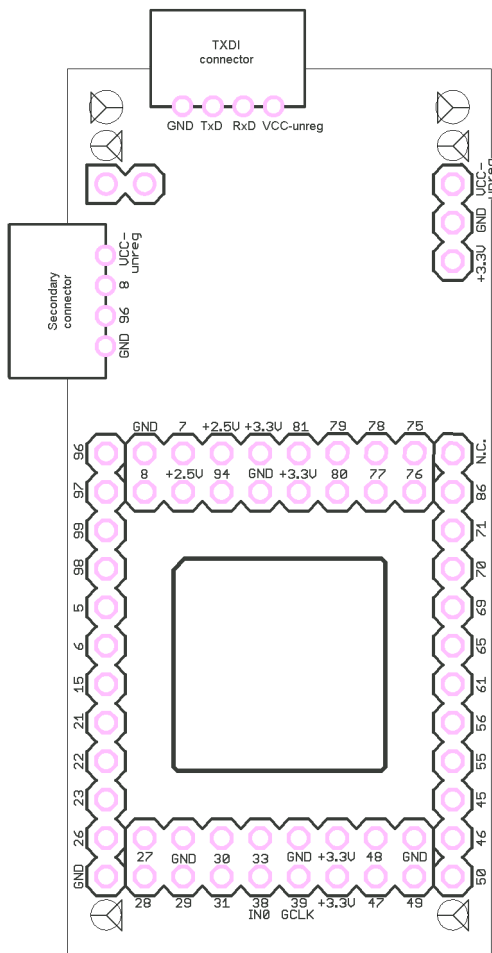
char ReadComChar()
{
    DWORD nRec;
    char c;
    if(!ReadFile(hCom, &c, 1, &nRec, NULL)) exit(1);

    return nRec ? c : 0;
}

void main()
{
    OpenCom("COM1:"); // change that to use a different COM port
    WriteComChar(0x41);
    CloseCom();
}
```

# 10 Board dimensions and IO pin assignment

## 10.1 Pluto



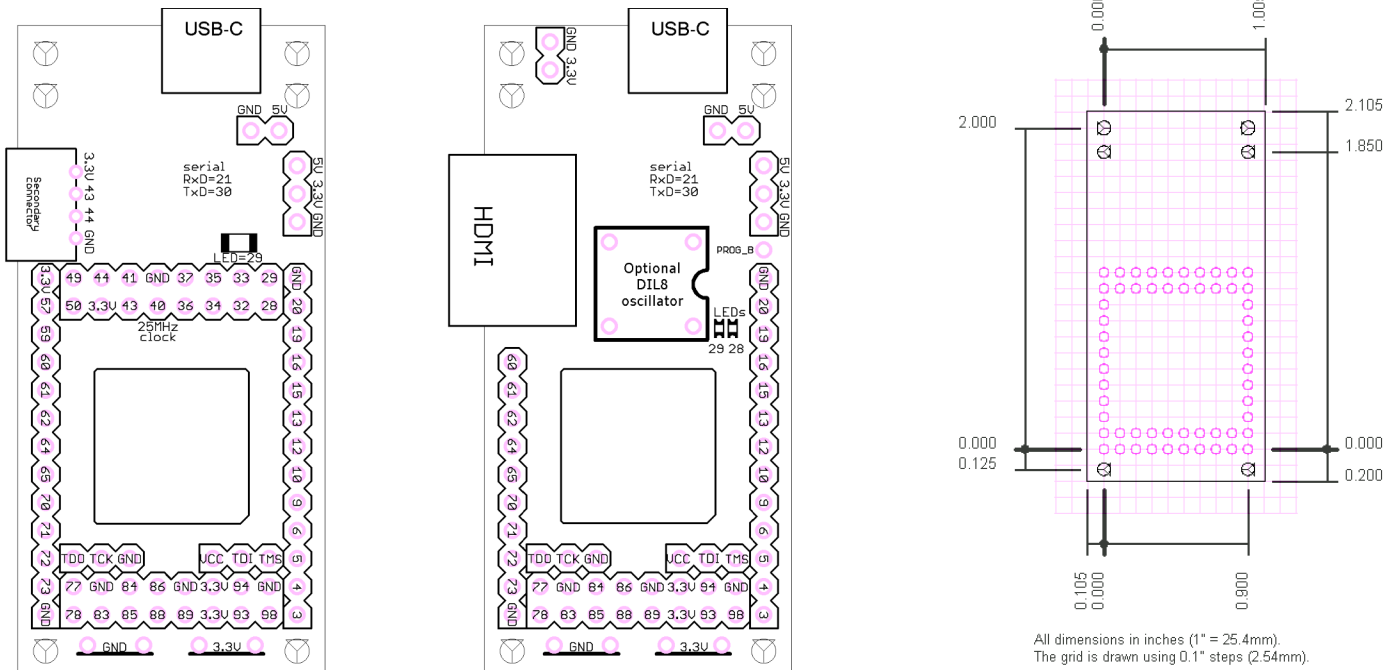
All IOs use 3.3V powered banks and are 5V input tolerant.

Pluto has 39+2 IOs available (the +2 are one clock and one dedicated input). Pin 39 is a dedicated clock input. Pin 38 can also be used as a clock input.

Check the [ACEX family datasheet](#) for more details.



## 10.3 Pluto-Ilx / HDMI



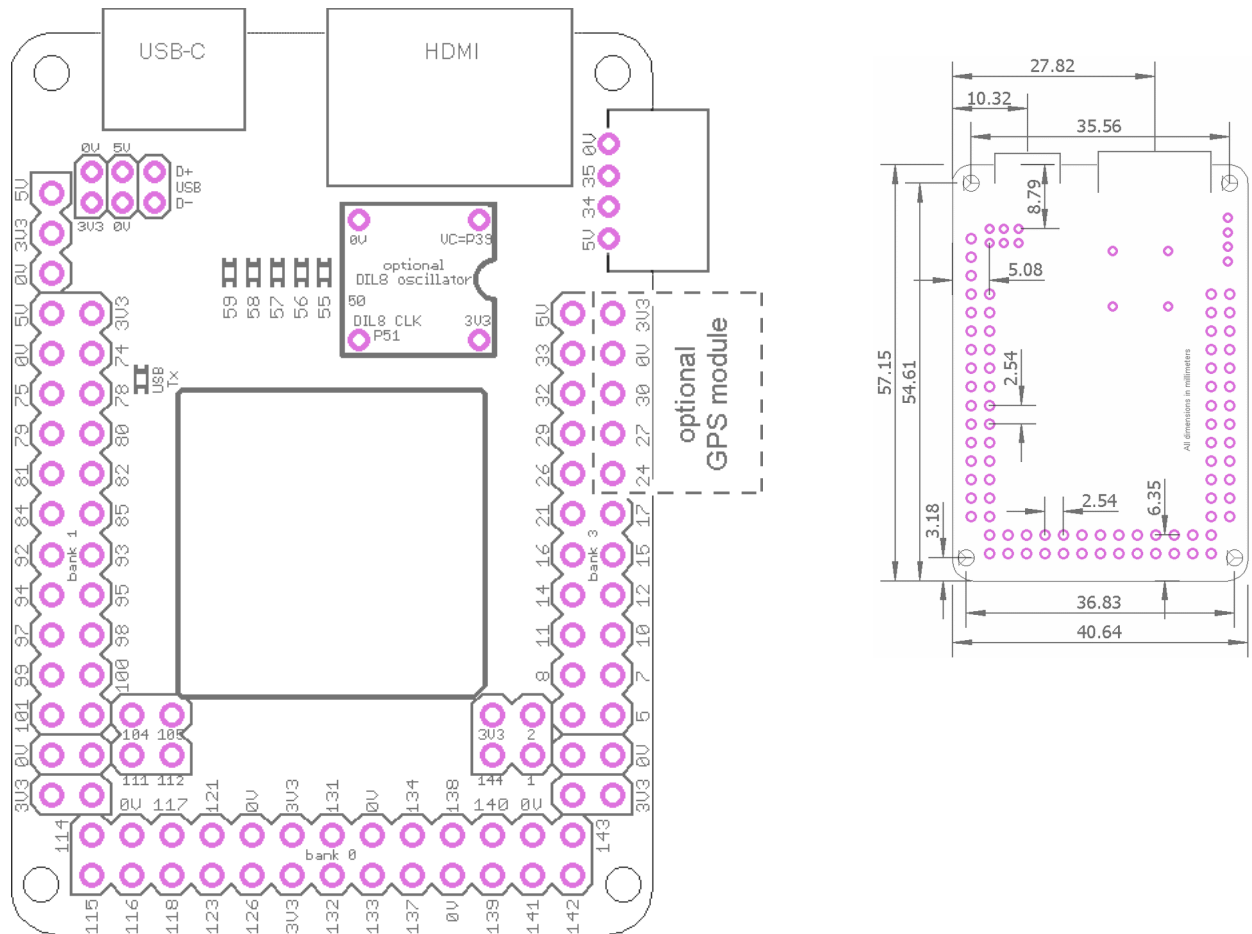
All IOs use 3.3V powered banks and are not 5V tolerant.

The main oscillator (pin 40) is always enabled.

LHCLK pins 9, 10, 12, 13, 15, 16, RHCLK 59~62, 64, 65, and GCLK 83~86, 88, 89 can be used as additional clocks.

Check the [Spartan-3A user guide](#) for more details.

## 10.4 Pluto-XC6 HDMI



All IOs use 3.3V powered banks and are not 5V tolerant.

Many pins can be used as additional clocks: pins 14~17, 21, 24, 51 (optional DIL8 oscillator), pin 88 (optional SMD oscillator at the bottom of the board, can also be fitted with a VCTCXO), pins 92~95.

The USB signals are available on a 2mm pitch header below the USB-C connector.

An extra secondary connector header with 3.3V power is available at the bottom of the board below the FPGA.

A few extra IOs are available on PTH and SMD test pads around the FPGA.

Check also the [Spartan-6 family overview](#) and the [Spartan-6 datasheet](#).

