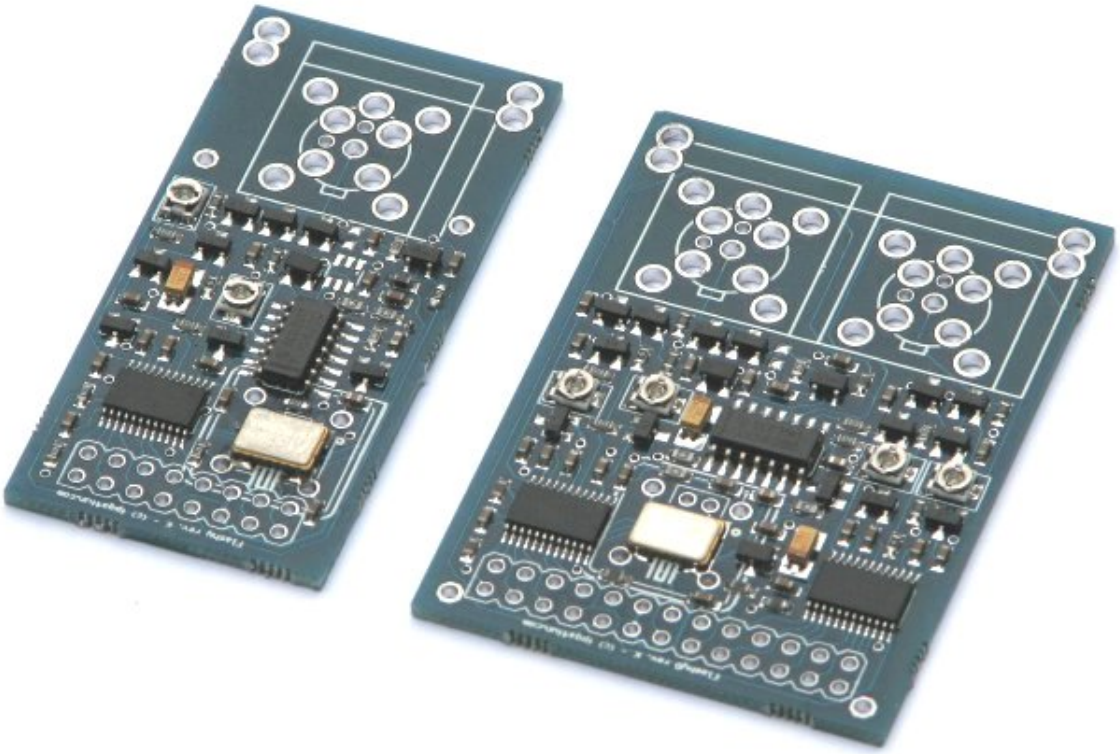
KNJN Flashy acquisition boards

http://www.knjn.com/



Document last revision on **October 20, 2017**

Table of Contents

1 Introduction

1.1 What is Flashy?

Flashy is an analog-to-digital converter (ADC) board. It is commonly used with an FPGA board to create a digital oscilloscope.

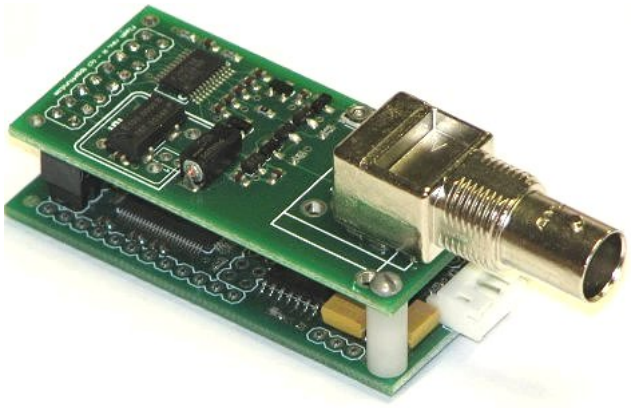
Flashy is a generic name used along this document that covers the Flash, Flashy, FlashD and FlashyD boards.

1.2 Capabilities

- 8-bit @ 60MSPS, 100MSPS or 200MSPS (based on National Semiconductor ADC08060/08100/08200).
- One or two Input stage(s)
  - 100MHz bandwidth (typ.)
  - DC-coupled 1MΩ high impedance input, regular oscilloscope probe compatible.
  - Accepts different HF connectors (BNC, SMA, RCA).
- Period output option.
- Clocked using an on-board oscillator, or from an outside source.
- 3.3V digital outputs.
- 0.1" (2.54mm) spacing output header.

1.3 Power requirements

- The Flashy boards are powered using 3.3V
- Flash/Flashy (one channel) consume about 100mA, while FlashD/FlashyD (two channels) consume about 200mA.

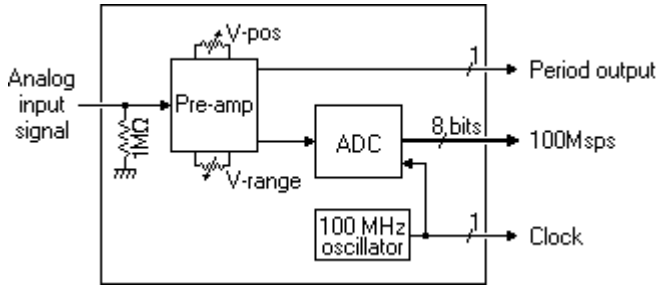


A simple PC-based one-channel oscilloscope: a Flash board (top) and a KNJN Pluto-II FPGA board (bottom).

2 How it works

2.1 Flashy

Here's the block diagram of a typical Flashy board.

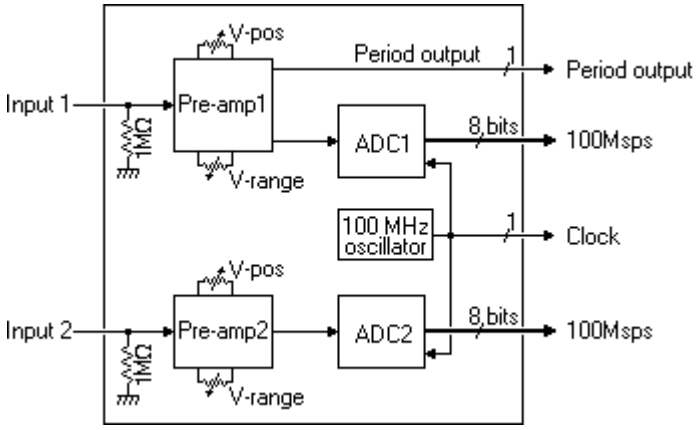


The input on the left is analog, while the outputs on the right are digital.

- The analog input signal is adjusted by the pre-amplifier and then fed to the ADC which digitizes the signal and creates an 8-bit 100Msps output.
- The ADC is clocked from the 100MHz local oscillator. The 100MHz clock signal is also made available outside Flashy, so that the 8-bit output can be synchronously captured.
- The V-pos and V-range controls allow moving the signal up or down (vertical scale).
- For the horizontal scale, the ADC is always clocked at 100MHz. If a different time base is desired, a digital filter down-samples the data (usually in an FPGA board attached to Flashy).
- The period output allows measuring the frequency of the input signal and reconstructing periodic signals (a technique called "equivalent-time sampling", see chapter 5).

2.2 FlashyD

FlashyD is the two inputs version of Flashy.



2.3 Flash and FlashD

Flash and FlashD are a low-cost version of Flashy and FlashyD. They lack the period output and the V-range control.

3 Flashy connection

3.1 FPGA boards

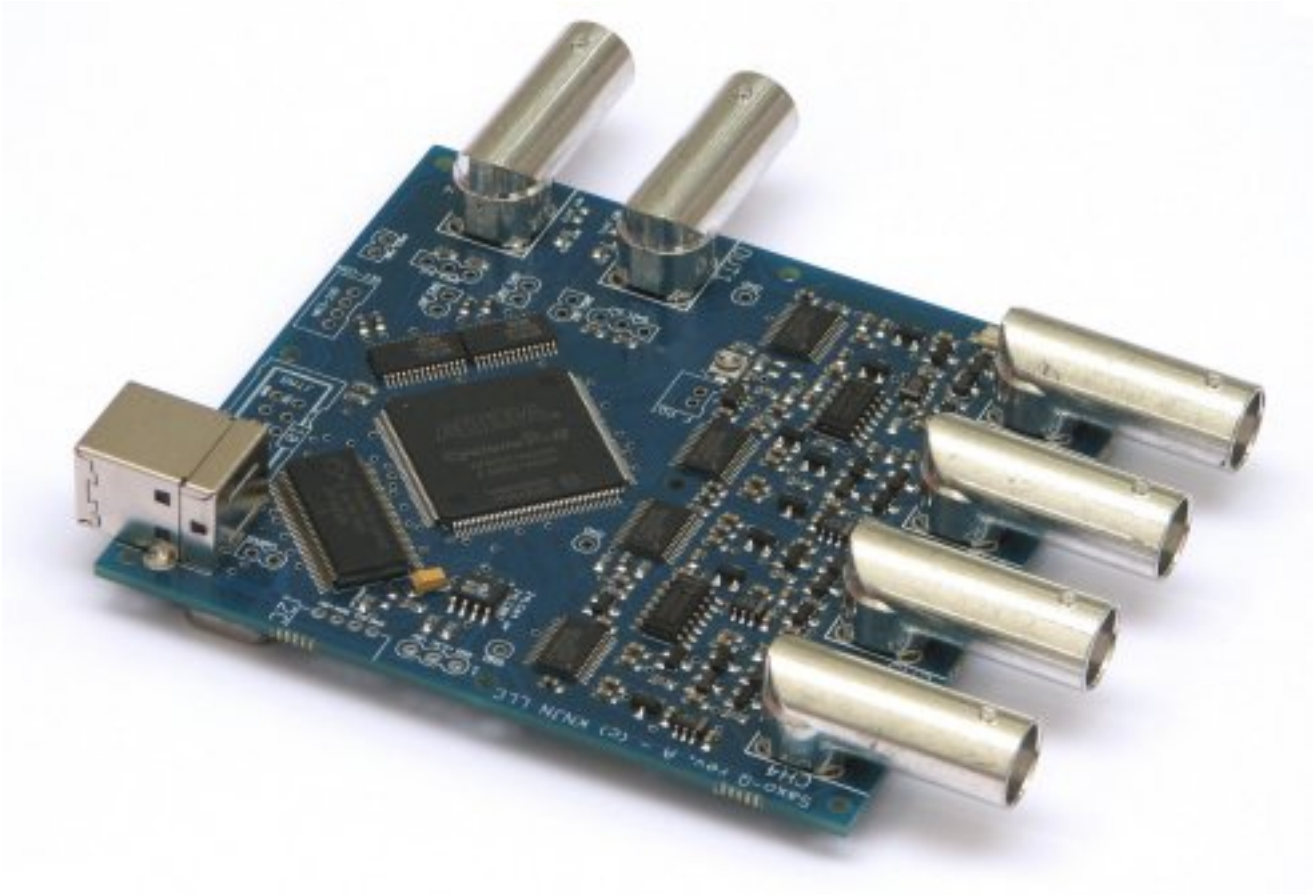
Flashy must be connected to another board for data storage, processing and display. Flashy is too fast for a direct connection to a CPU but can be used in conjunction with an FPGA board that acquires the data at high-speed and re-transmits it to a PC at a lower speed. The PC can then process/display the data.

- Flashy is compatible with all KNJN FPGA boards. KNJN FPGA boards are provided with demonstration projects named FlashyDemo and FlashyMini (see chapters 4 and 6).
- Flashy can be connected to other FPGA boards as well (3.3V power and 3.3V logic signals).

Note that boards that accept FlashD/FlashyD can also accept Flash/Flashy (Flash/Flashy pinout is a subset of FlashD/FlashyD).

3.2 Saxo-Q

Saxo-Q is a particular KNJN FPGA board with the equivalent of four Flashys on-board (i.e. four high-speed analog inputs). It also has two high-speed analog outputs.



Saxo-Q

4 FlashyDemo

4.1 What is it?

FlashyDemo is a design provided with all KNJN FPGA boards. It allows using Flashy as a digital oscilloscope with advanced features like:

- Multiple traces persistence
- Equivalent time sampling

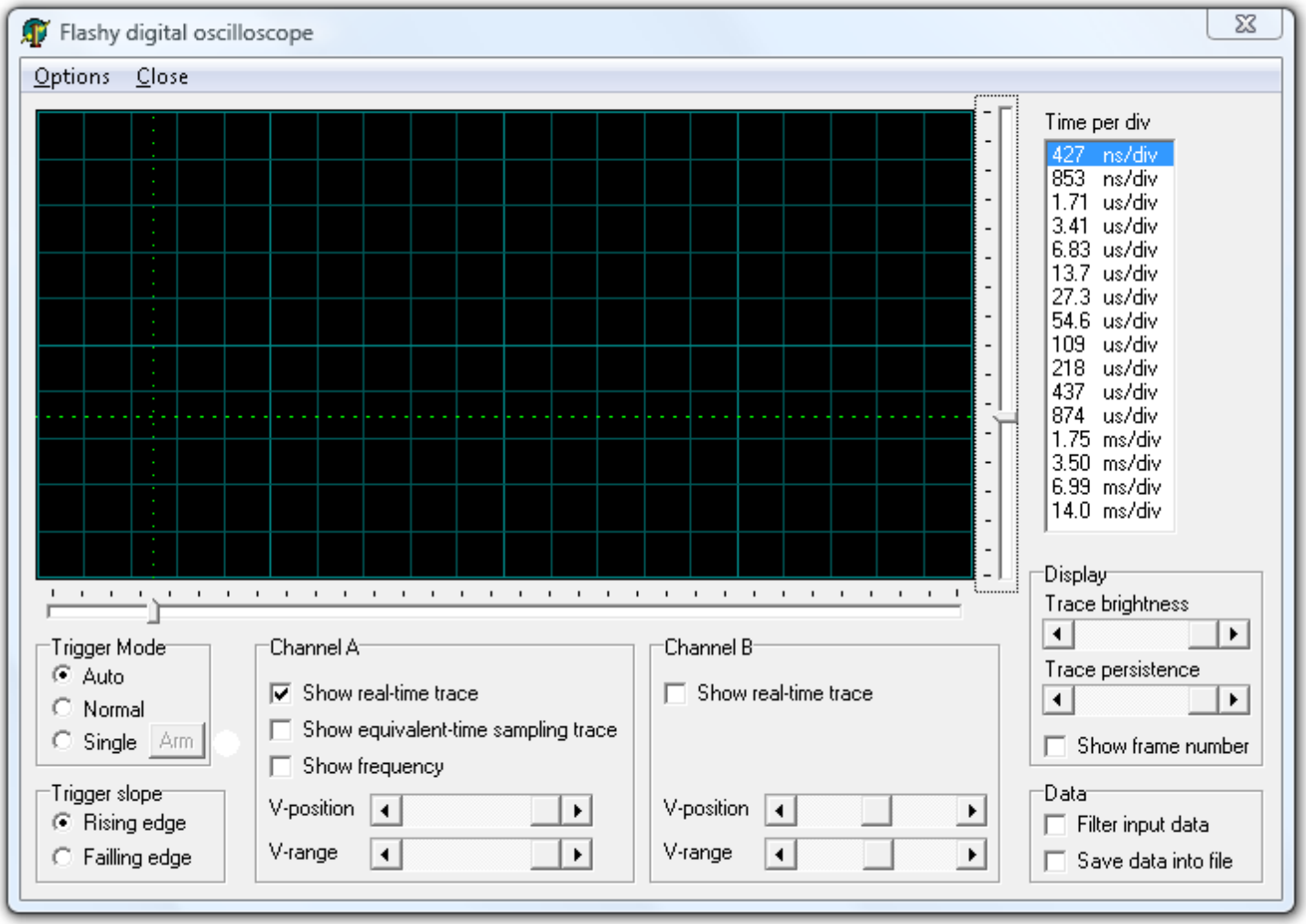
The number of channels displayed is automatically adjusted to the board you are using: one channel for Flash/Flashy, two channels for FlashyD and four channels for Saxo-Q.

4.2 Get the oscilloscope window

Follow this procedure:

1. Open FPGAconf and configure your FPGA with the “FlashyDemo.rbf” or “FlashyDemo.bit” provided with your KNJN FPGA board.
2. Press CTRL-F.

The FlashyDemo window appears and the oscilloscope is ready to work.

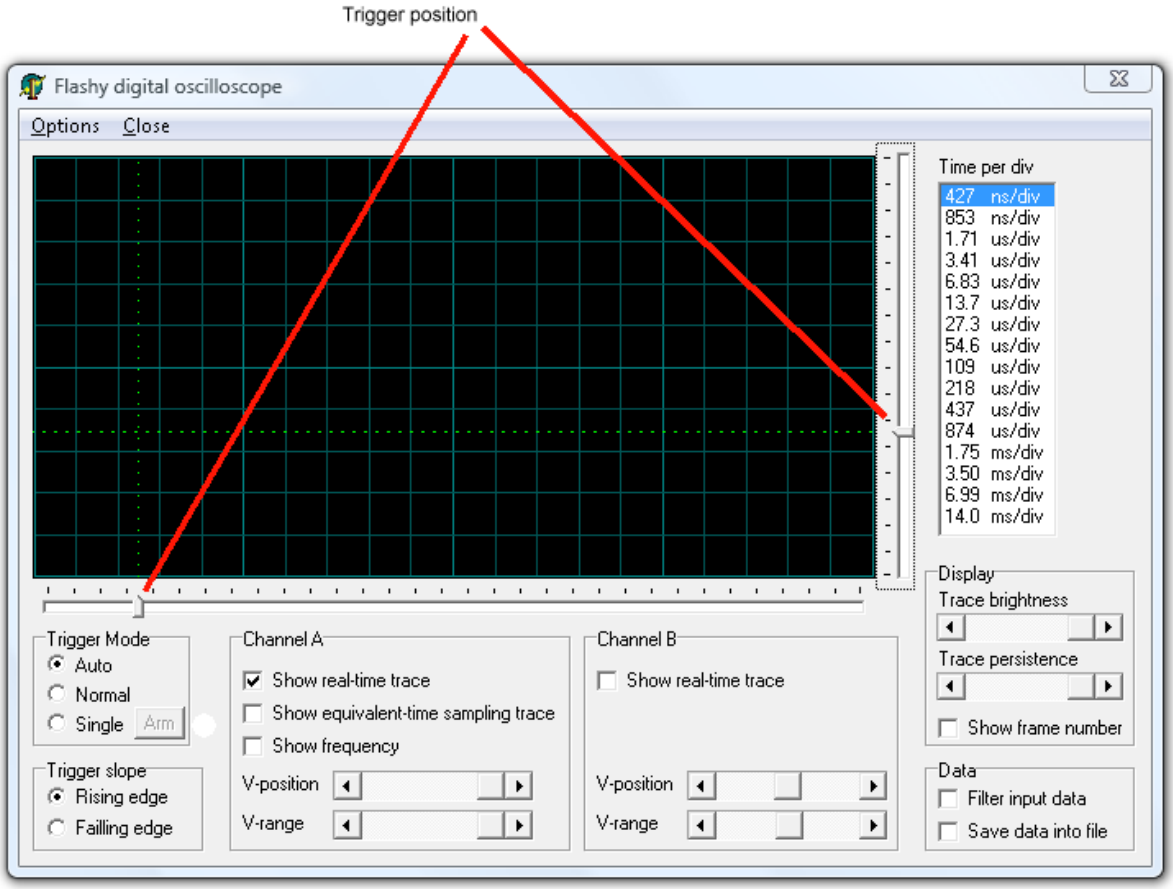


4.3 GUI controls

The FlashyDemo GUI provides access to the:

- Trigger mode (auto, normal, single), trigger slope (rising, falling), trigger position
- Acquisition frequency (time per div.)
- Trace brightness & persistence

The trigger levels can be adjusted using sliding bars.

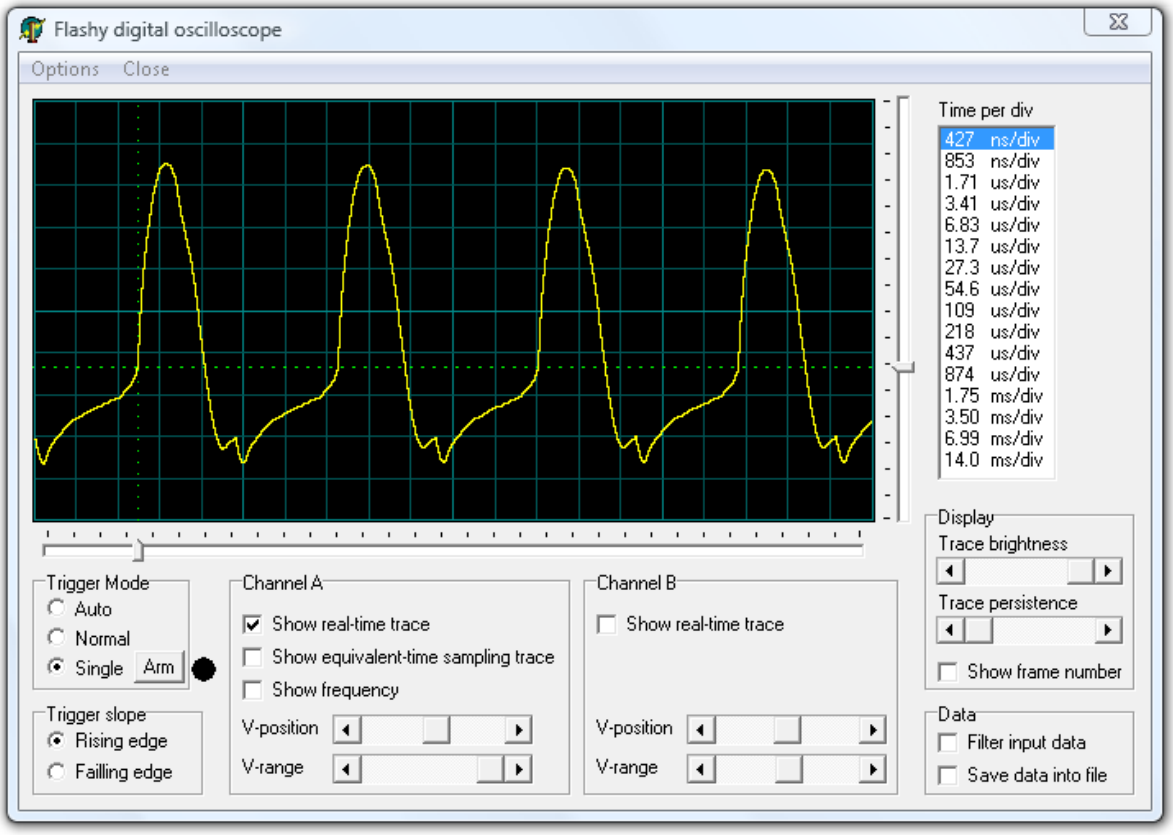


Some interesting features:

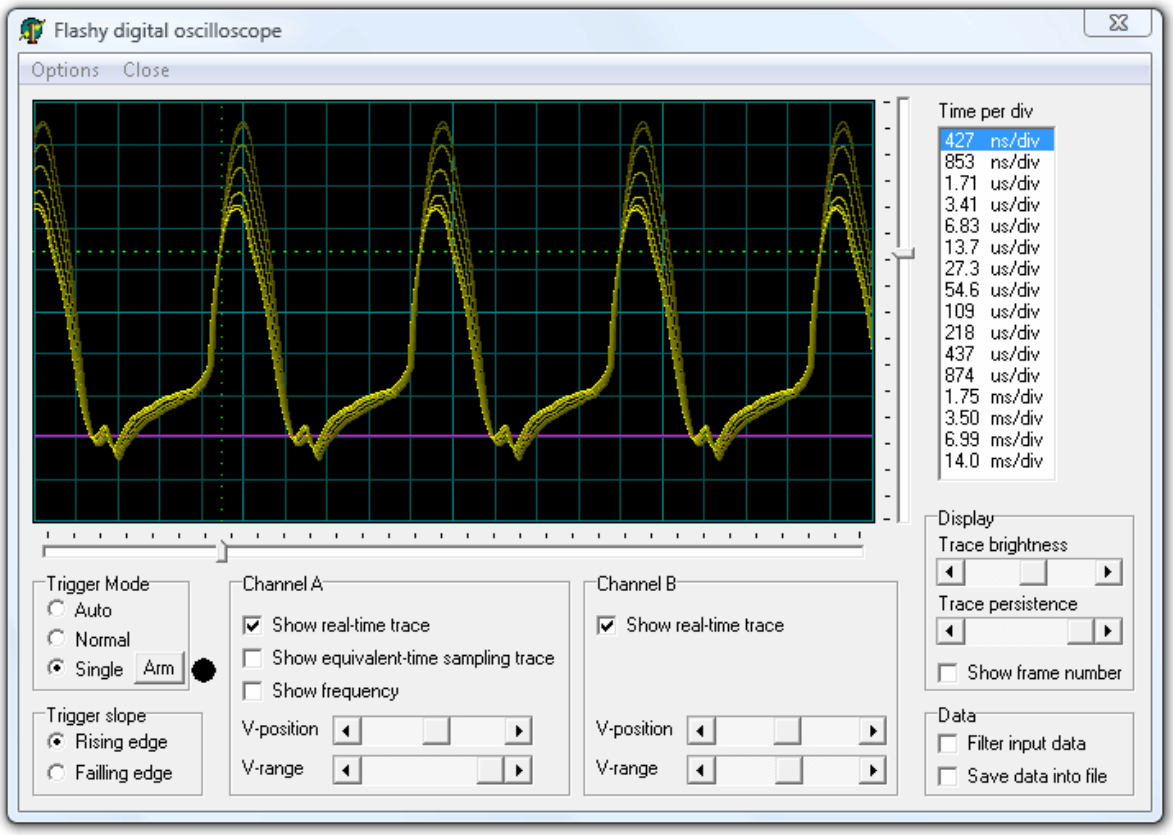
- The “single” trigger mode allows freezing a running trace at any time and start individual acquisitions.
- The “Save data into file” saves all the trace data into a binary file (for post-acquisition analysis).

4.4 Trace persistence

Here's an example of acquisition where trace persistence is off:



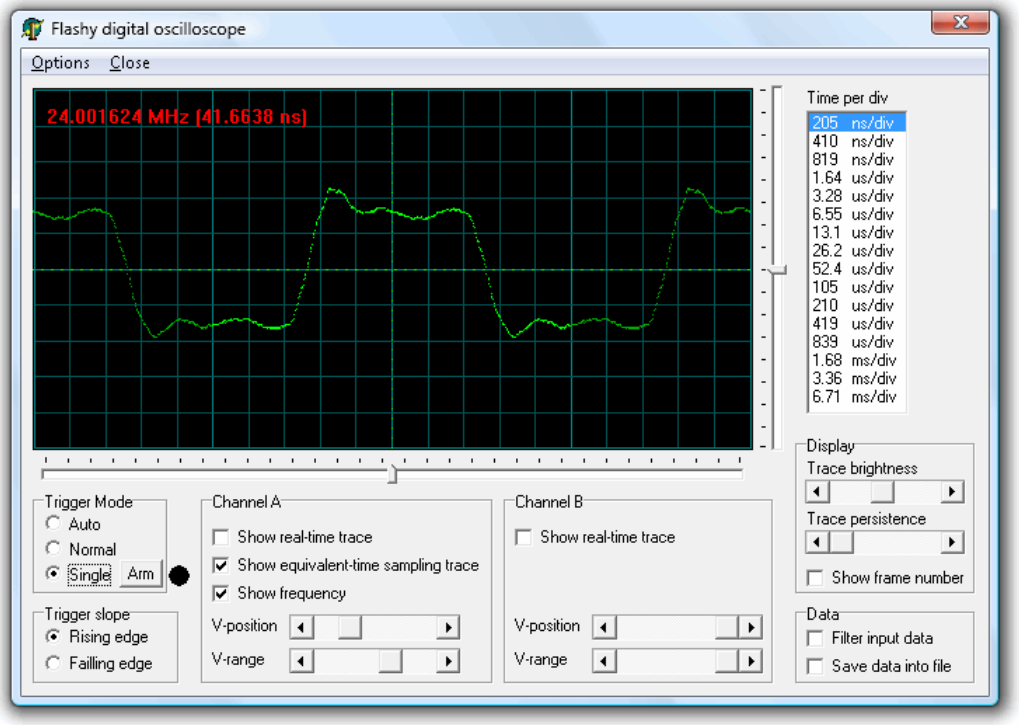
Now we enable the trace persistence (with a signal decreasing in amplitude) and also channel B (with no signal):



4.5 Frequency meter

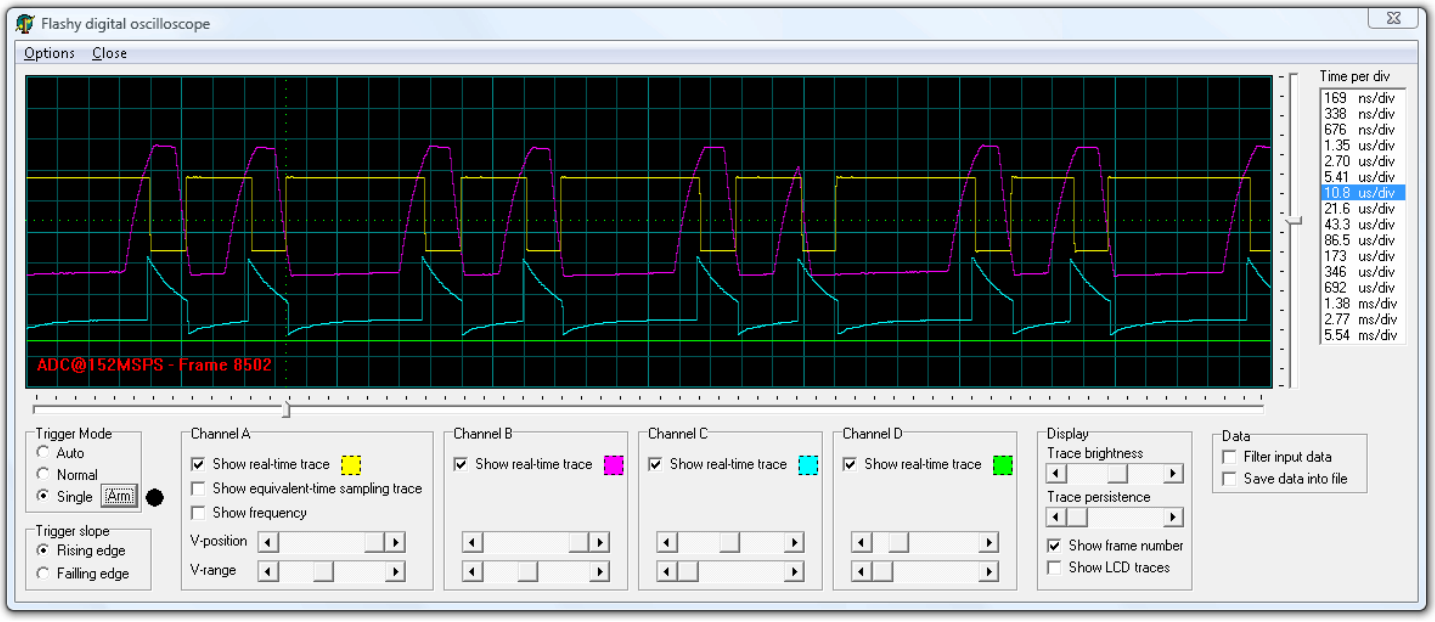
If your Flashy board has the period output option, FlashyDemo can measure the frequency of the incoming signal.

This also allows working in equivalent time sampling mode (chapter 5). For example, here's a 24MHz signal sampled at 100MSPS with equivalent time sampling on.



4.6 4 channels version

Starting with FPGAconf 2.11, FlashyDemo supports up to four channels and a larger acquisition window.



4.7 Protocol

You can use your own software to acquire data, see the FlashyDemo protocol (chapter 13).

4.8 Pluto limitations

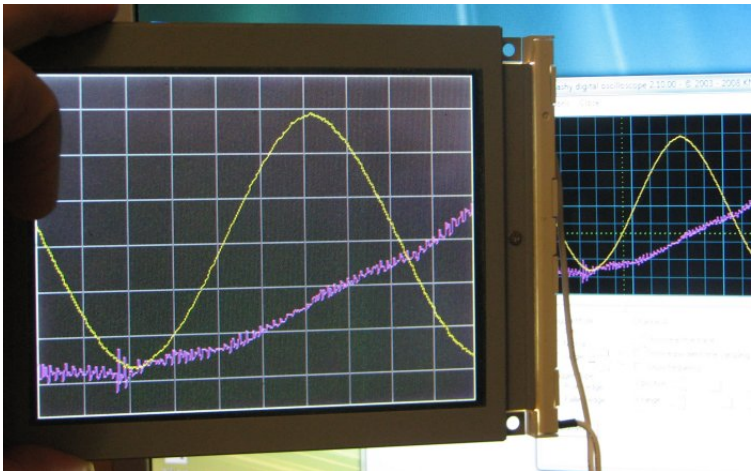
FlashyDemo has some limitations when used with a Pluto FPGA board (an entry-level KNJN FPGA board) and a Flashy rev. J (or subsequent revisions).

- The DAC control resolution is limited
- The sampling rates are limited.

This is due to Pluto's small FPGA which cannot accommodate all the functionality. No other KNJN board (including Pluto-II/-IIx and Pluto-3) have these limitations.

4.9 External LCD

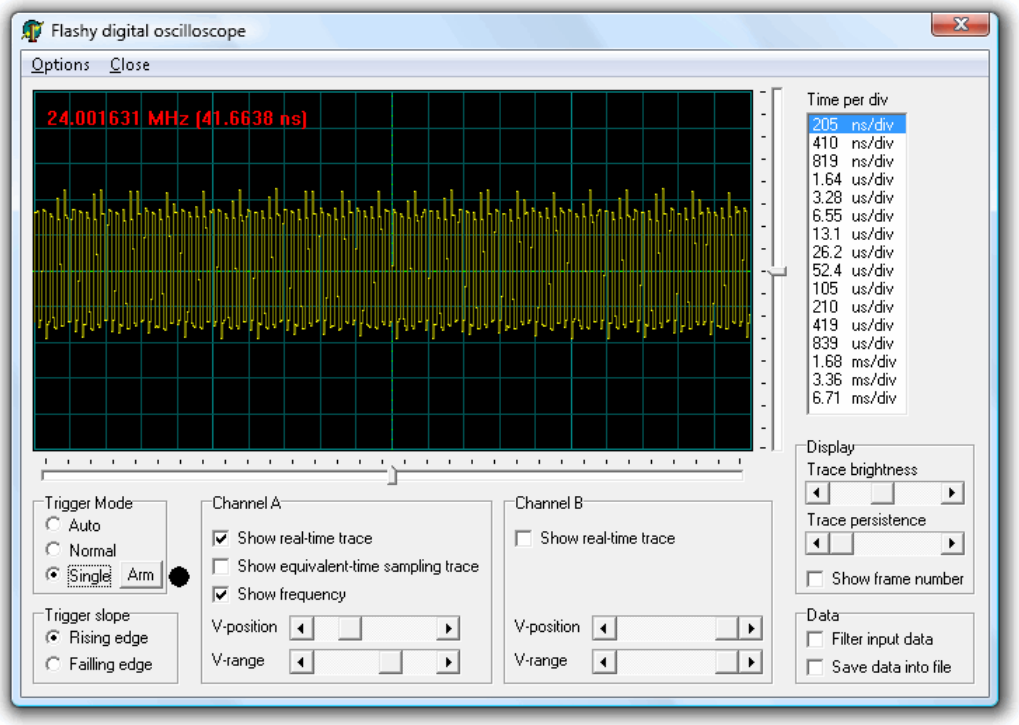
FlashyDemo can drive an external color LCD (KNJN item#5300) when used with the following KNJN FPGA boards: Pluto-II rev. E and above, Pluto-3, Saxo/Xylo rev. E and above, Saxo-L rev. B and above, Xylo-EM, Xylo-L, Xylo-LM, Saxo-Q.



5 Equivalent-time sampling

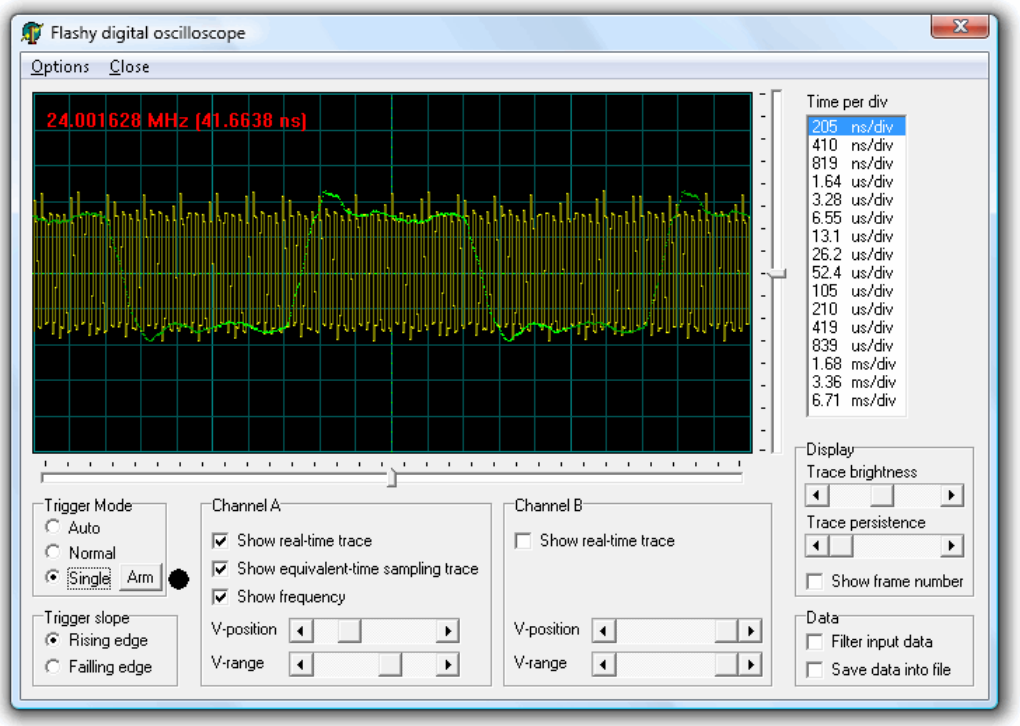
5.1 How it works

FlashyDemo has an interesting capability when probing high-speed periodic signals: “equivalent-time sampling”. Let's see how it works with an example. Here we probe a 24MHz signal using a 100MHz Flashy board.

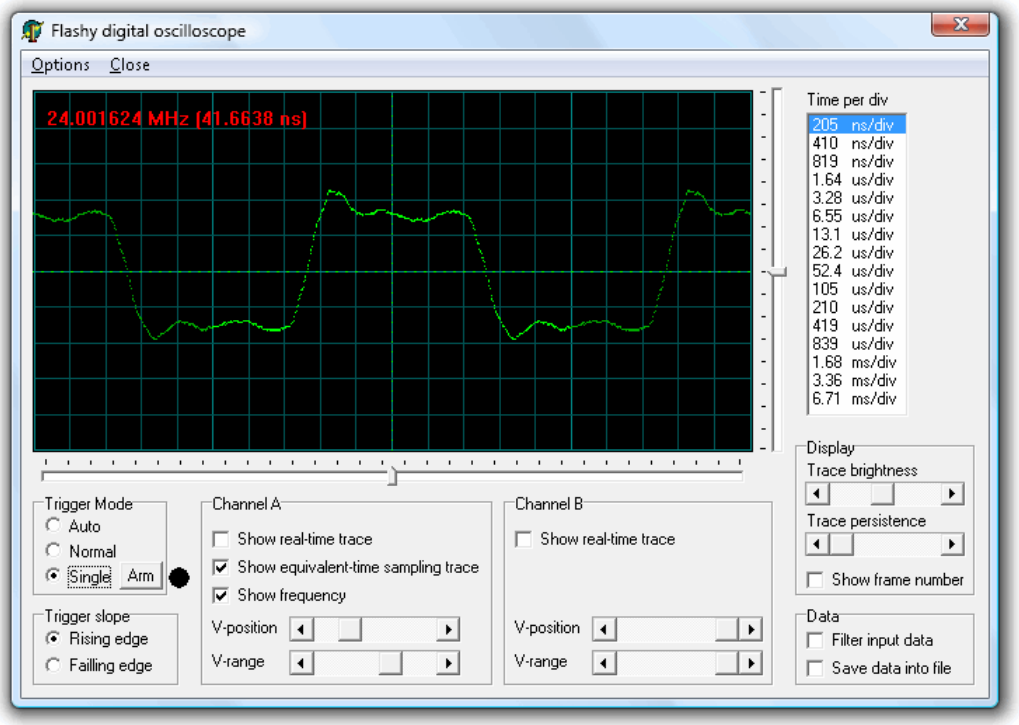


The signal is hardly recognizable. But it is periodic and its frequency is known thanks to Flashy's period output, so it is possible to “reconstruct” it.

First, click on “equivalent time sampling” to get a green waveform added to the display (mixed with the yellow channel A trace).



Now disable the channel A trace to isolate the equivalent-time sampling 24MHz signal.



5.2 Limitations

For the equivalent-time sampling to work, the following requirements need to be met:

1. The input signal is periodic.
2. You use a Flashy/FlashyD board with period output, and the frequency is acquired/displayed correctly.
3. The input signal frequency is not a multiple of the sampling frequency.
   For example, if you probe a 25MHz signal sampled at 100MHz, the reconstruction routine has very little to work with (the same 4 points per periods) and cannot reconstruct much of the input signal (it just shows 4 dots on the screen).

Notes:

1. The “Time per div” controls the sampling of the signal (yellow trace). The green trace is the reconstructed signal, and shows one period of the signal, regardless of the “Time per div” setting.
2. Unlike real-time sampling, the input signal can be above 50MHz (the “Nyquist frequency”) and even above 100MHz for the reconstruction to work.
3. As the input signal frequency goes up, the reconstructed signal amplitude is reduced by the finite bandwidth of the Flashy input stage.

6 FlashyMini

6.1 What is it?

FlashyMini is a design provided with most KNJN FPGA boards. It demonstrates a simple way to acquire the Flashy ADC data at high-speed and transmit it to a PC.

6.2 Features

FlashyMini is provided in source code form:

- HDL design (runs in the FPGA)
- C code (runs in the PC).

The code can be studied and used as a starting point for more complicated projects. FlashyMini architecture was kept as simple as possible to make the source code easy to understand:

- Only one channel is transmitted to the PC
- No trigger mechanism is implemented
- No down-sampling (i.e. full ADC speed acquisition @ 100MHz typical)

6.3 How it works

The FPGA gets data from Flashy at high speed (100MHz typical). The data stream fills quickly a 512 bytes blockram in the FPGA. The acquisition is stopped once the blockram is full. The data is sent to the PC (at a lower speed) and the process starts over.

6.4 HDL compilation

Use Altera's Quartus-II or Xilinx's ISE to compile the FlashyMini HDL project provided with your KNJN board. Upon compilation, you get an FPGA bitfile.

6.5 C compilation

Compile the C code. Note that many different C compilers can be used, including Microsoft Visual C++. You get an exe file.

6.6 Run FlashyMini

You are now ready to run FlashyMini:

1. Configure the FPGA with the bitfile generated in step 6.4
2. Run the exe file compiled in step 6.5

The code runs in a command-line window and displays the numerical values of the signal digitized by Flashy.

Once this works, modify the code for your own needs.

7 Analog input

Flashy can accept analog signals directly, or through an oscilloscope probe.

7.1 Oscilloscope probe

When Flashy is fitted with a BNC connector (chapter 9), it can be used with common oscilloscope probes.

Most probes are 10:1 (or switchable 10:1/1:1). The probe setting determines the Flashy input range.

Probe	*Typical input range*	*DUT load*
10:1	-1V to +10V	10MΩ
1:1	-100mV to +1000mV	1MΩ

A 10:1 probe is usually the best choice for probing digital logic signals, because it provides an adequate input range and a very low load on the device under test (DUT).

7.2 Direct input

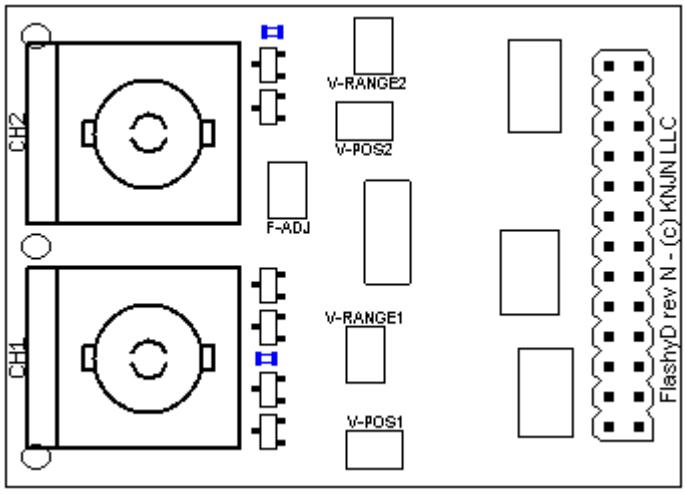
If you are feeding a signal directly to Flashy (i.e. you are not using an oscilloscope probe), the input range is the same as with a 1:1 probe.

Mode	*Typical input range*	*Load*
Direct input	-100mV to +1000mV	1MΩ / 15pF

7.3 Input coupling/impedance

Flashy has DC-coupled high impedance 1MΩ inputs by default. The impedance can be lowered to 50Ω or 75Ω (to acquire video signals for example) by changing one resistor per board input stage.

For example, here's the location of the resistors for FlashyD rev. N



7.4 Extended dynamic range

To go beyond Flashy's native limits, use a resistor divider (or a potentiometer) to reduce the voltage seen by Flashy. This is easy thanks to Flashy's native high impedance input stage (1MΩ).

For example, a 1KΩ/9KΩ resistor divider can be used to divide by 10. With a 10:1 probe, that allows signals between -50V to +100V to be probed.

Flashy's latest revisions allow putting a resistor divider right on the board.

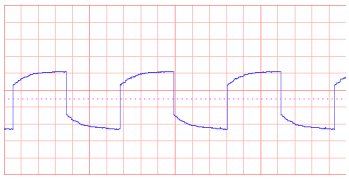
8 Oscilloscope probe adjustment

8.1 Adjustment procedure

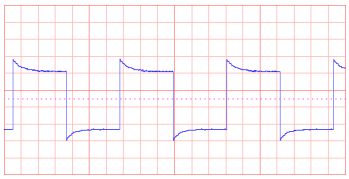
Every oscilloscope probe has a “low-frequency compensation” control that needs to be matched to the input it is used with.

The first time you use a probe with Flashy, please follow this procedure:

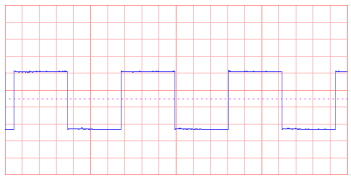
1. Feed the probe with a square wave signal of about 1 Khz.
2. Set the oscilloscope horizontal control speed to about 1ms/div.
3. Adjust the “low-frequency compensation” on the probe to get a square wave output.



Under-compensated



Over-compensated



Good

8.2 Multiple channels

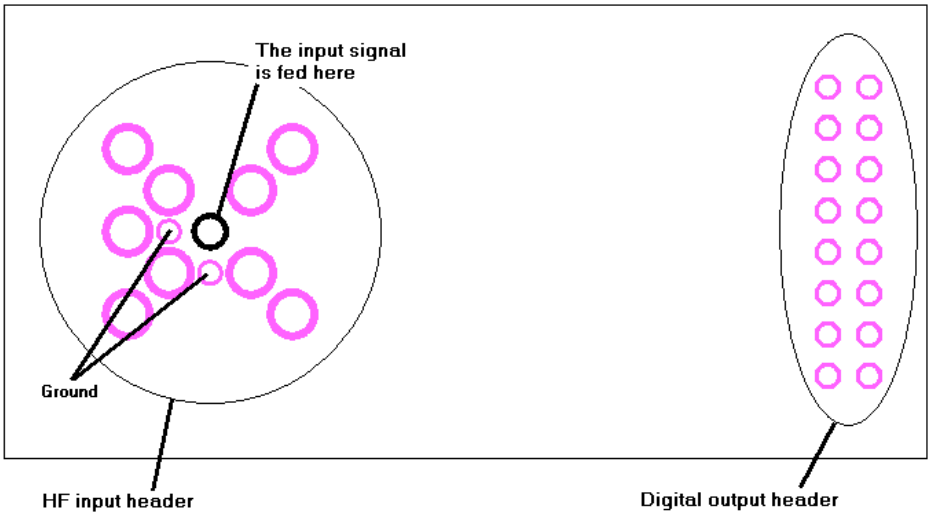
With FlashyD, the adjustment has to take place for each probe/channel pair.

9 HF connector

9.1 HF input header layout

Flashy accepts many types of HF connectors, including BNC, SMA and RCA.

Here's a view of Flashy's headers.



The HF input header is on the left, the digital output header on the right.

1. The most common HF connector is a BNC, straight or right-angle.
2. The input signal is fed in the middle through-hole, with ground on one of the two small through-holes next to it. The other through-holes may or may not be grounds.

9.2 Right-angle BNC on Flash/Flashy

You have the two mounting directions choices.

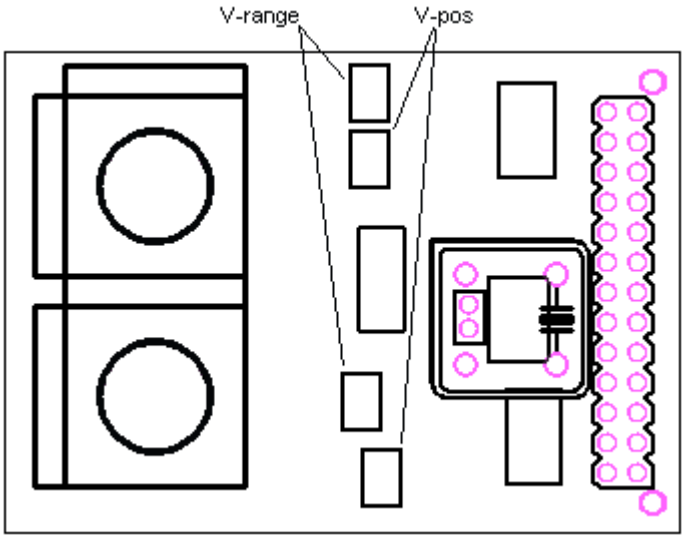
Illustration 1: BNC right-angle mounting position 1

Illustration 2: BNC right-angle mounting position 2

10 Flashy potentiometers

10.1 Layout

Flashy has two potentiometers per channel to control V-range and V-pos.
For example, here's FlashyD rev. K:



10.2 Period output

Another potentiometer may be available on certain board revisions for fine control of the period output (allows the FPGA to measure the frequency of the input signal on channel A).

- The period output function works best on square or sine input signals. Then adjust the control to get a fixed and stable frequency reading.
- The adjustment is sensitive; use if possible a non-metallic screwdriver (the adjustment screwdriver that comes with oscilloscope probes is usually adequate).
- The period output works in the range 100KHz to 100MHz (typ.)

10.3 Potentiometers and DACs

In addition to the potentiometers, certain board revisions have DACs that share control of V-range and V-pos.

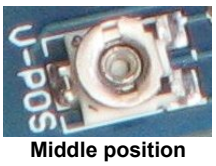
- When the DACs are initialized (see chapter 14), V-pos range is the sum of the DAC setting and pot setting, while V-range is the maximum of the DAC setting and the pot setting.
- When the DAC is not initialized, the pots are in full control.

Control	Effect
V-pos	Pot + DAC
V-range	max(Pot, DAC)

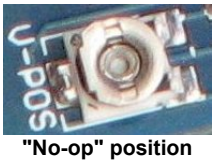
10.4 Potentiometer positions

If you want to control a board only with the DACs, it is usually required that you turn the potentiometers in a “no operation” (“no-op”) position. To get into “no-op”, turn the round part of the potentiometer between the two end pins.

Note: the boards are shipped in the “no-op” position to allow DAC control.



Middle position



"No-op" position

11 Local oscillator

11.1 Local oscillator

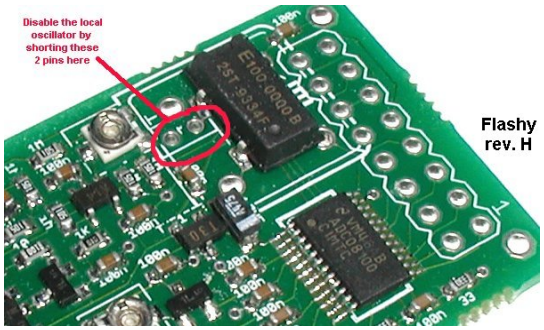
Flashy is shipped with a local oscillator from 60 to 133MHz (depending on the model). The oscillator is usually an SMD, but a “half-can” 3.3V DIL8 oscillator may also be used.

Flashy’s ADC can run at different speeds, see the ADC08060/ADC08100/ADC08200 datasheets for details.

11.2 Disabling the local oscillator

You might want to disable the local oscillator to use other oscillators, or an external clock source.

Typically, the local oscillator may be disabled by tying its pin 1 to ground. Prior to Flashy rev. H, this was accomplished by soldering a 0 ohm resistor next to the oscillator pin 1. Flashy rev. H and above can be fitted with a 2mm jumper (or by shorting the 2 pins header) to disable the local oscillator.



Note: Some oscillators do not go into tri-state when disabled. In this case, the oscillator has to be de-soldered from the board before another clock source can be used.

12 ADC output header

12.1 Header

Flashy uses a standard 0.1” (2.54mm) pitch header.

All signals are high-speed logic signals and may require termination.

12.2 Pinout (rev. H and above)

Flash/Flashy has a 2x8 header.

DA6 (pin 2)	GND	DA3	DA2	GND	VCC (3.3V)	DA1	GND (pin 16)
DA7 (pin 1)	DA5	DA4	CLK	Period OutA	VCC (3.3V)	DA0	DAC control (pin 15)

FlashyD has a 2x13 header.

DA6 (pin 2)	GND	DA3	DA2	GND	VCC (3.3V)	DA1	GND (pin 16)	DB7	DB6	DB4	GND	DB1 (pin 26)
DA7 (pin 1)	DA5	DA4	CLK	Period OutA	VCC (3.3V)	DA0	DAC control (pin 15)	DB5	GND	DB3	DB2	DB0 (pin 25)

- DA7-0 is the 8-bit ADC output for channel A.
- DB7-0 is the 8-bit ADC output for channel B (FlashD/FlashyD only).
- CLK is the clock signal output generated by Flash/Flashy/FlashD/FlashyD. The DA and DB buses are synchronous to CLK. If the Flashy on-board oscillator is removed or disabled, the CLK signal becomes an input.
- VCC is the board power (3.3V), GND is the board ground (0V).
- “Period OutA” (pin 9) is an output that toggles at each signal period of channel A (see 10.2).
- “DAC control” (pin 15) is an input and is used only on Flashy/FlashyD (see chapter 14).

The range of clock frequencies allowed are listed in the ADC datasheets. They are repeated below for your convenience.

- ADC08060 runs from 20MHz to 70MHz typ.
- ADC08100 runs from 20MHz to 125MHz typ.
- ADC08200 runs from 10MHz to 230MHz typ.

13 FlashyDemo protocol

The protocol described here is used in FPGAconf versions 2.09.xx to 2.11.xx

13.1 Protocol

The FlashyDemo protocol is simple:

1. You send a trigger request to the FPGA
2. You receive a data packet back.

13.2 Trigger request

The trigger request is an 8 byte packet for 2.09.xx, 10 byte packet for 2.10.xx (two extra bytes as prefix), and 14 byte for 2.11.xx (four extra DAC control bytes).

Offset	Bits	Name	Value	FPGAconf version
		Prefix	Two null bytes	2.10.xx +
0	7:0	DAC_reg[0]	V-range for channel 0	2.09.xx +
1	7:0	DAC_reg[1]	V-pos for channel 0	2.09.xx +
2	7:0	DAC_reg[2]	V-range for channel 1	2.09.xx +
3	7:0	DAC_reg[3]	V-pos for channel 1	2.09.xx +
4	7:0	DAC_reg[4]	V-range for channel 2	2.11.xx +
5	7:0	DAC_reg[5]	V-pos for channel 2	2.11.xx +
6	7:0	DAC_reg[6]	V-range for channel 3	2.11.xx +
7	7:0	DAC_reg[7]	V-pos for channel 3	2.11.xx +
8	7:0	TriggerThreshold	Vertical trigger value (1 to 255, and 0 means trigger now)	2.09.xx +
9	7:0	PreTriggerPoint	Horizontal pre-trigger point (4 to 252)	2.09.xx +
10	3:0	HDiv	Horizontal time base (0 to 15, 0=fastest, 15 = slowest)	2.09.xx +
	4	FilterDataIn	Data down-sampling (0=off, 1=on)	2.09.xx +
11	6	TriggerSlope	0=rising edge, 1=falling edge trigger	2.09.xx +
	7	Acq_Allowed	1=arm trigger	2.09.xx +

13.3 Data packet

Once the trigger request is sent, you can expect a data packet as soon as the trigger activates.

The data packet structure is:

```
#define nFlashyChannels 2            // 1 of Flashy, 2 for FlashyD, 4 for Saxo-Q
#define ChDataLen     512 or 1024    // 512 for FPGAconf up to 2.10.xx, or 1024 for FPGAconf 2.11.xx

struct DPH
{
    WORD magic_number;        // 0x55AA
    BYTE version;             // 0x01 for FPGAconf 2.10.xx, 0x02 for FPGAconf 2.11.xx
    BYTE nb_channels;         // 0x01 to 0x04 in low nibble (ignore high nibble)
};

struct
{
    struct DPH hdr1;                          // header
    BYTE samples[ChDataLen*nFlashyChannels]; // data acquired from Flashy
    DWORD dummy[4];                           // don't care data
    DWORD FC[3];                              // frequency counters
    struct DPH hdr2;                          // second header
} DataPacket;
```

If you own a KNJN FPGA board, check its startup-kit FlashyDemo directory for more information.

14 DAC control (Flashy and FlashyD)

14.1 DAC outputs

The DAC outputs functions are as follows:

DAC	Function	Number of bits	Note
DAC 0	V-range channel 0	6	
DAC 1	V-pos channel 0	7 or 8	
DAC 2	V-range channel 1	6	FlashyD only
DAC 3	V-pos channel 1	7 or 8	FlashyD only

Note: The range and position controls have a negative slope (the higher the value, the lower the range or position).

14.2 DAC control source code

The DACs are controlled using a single IO (the pin 15 “DAC control” signal on the ADC output header).

The following FPGA code can be used to drive the DAC control line.

```
// data to be programmed into up to 4 DACs (8 bit per DAC)
wire [7:0] DAC_reg [3:0];  // TO BE ASSIGNED

// In an FPGA, let's use a free running counter to update the DACs periodically
// If you are using a CPU, you could update the DAC only once (to save CPU cycles)

// With “clk” a free running clock (25MHz typical)
// let's create a counter
reg [31:0] DAC_cnt;  always @(posedge clk) DAC_cnt <= DAC_cnt + 1;

// now we put the DAC values into good format
// and set some bits to “1” to make sure some minimum values are observed
wire [7:0] DAC_regdata [3:0];
assign DAC_regdata[0] = ~DAC_reg[0] | 8'b11000000;
assign DAC_regdata[1] = ~DAC_reg[1] | 8'b10000000;
assign DAC_regdata[2] = ~DAC_reg[2] | 8'b11000000;
assign DAC_regdata[3] = ~DAC_reg[3] | 8'b10000000;

// create the Flashy DAC control signal
wire [15:0] DAC_data = {5'b11111, DAC_cnt[17:16], 1'b1, DAC_regdata[DAC_cnt[17:16]]};

// and send it
reg DAC_control;
always @(posedge clk)
       DAC_control <= &DAC_cnt[15:13] & (&DAC_cnt[8:1] ? ~DAC_cnt[0] : DAC_data[~DAC_cnt[12:9]]);
```

14.3 Extended DAC control

On Flashy revisions P and beyond, V-pos DAC control can use 8 bits instead of 7.

```
assign DAC_regdata[0] = ~DAC_reg[0] | 8'b11000000;
assign DAC_regdata[1] = ~DAC_reg[1];
assign DAC_regdata[2] = ~DAC_reg[2] | 8'b11000000;
assign DAC_regdata[3] = ~DAC_reg[3];
```

15 KNJN Dragon board errata

15.1 Dragon rev. C with FlashyD rev. H

FlashyD rev. H has a “no-connect” pin (pin 15) on its output connector. This needs to be tied to DA1 (pin 14) using a 0 ohm resistor, or a short wire.

See paragraph 12.2 for the connector pinout.

15.2 Dragon rev. C with FlashyD rev. J/K

Dragon rev. C cannot access DA1 of FlashyD rev. J, unless a wire is added on the board.

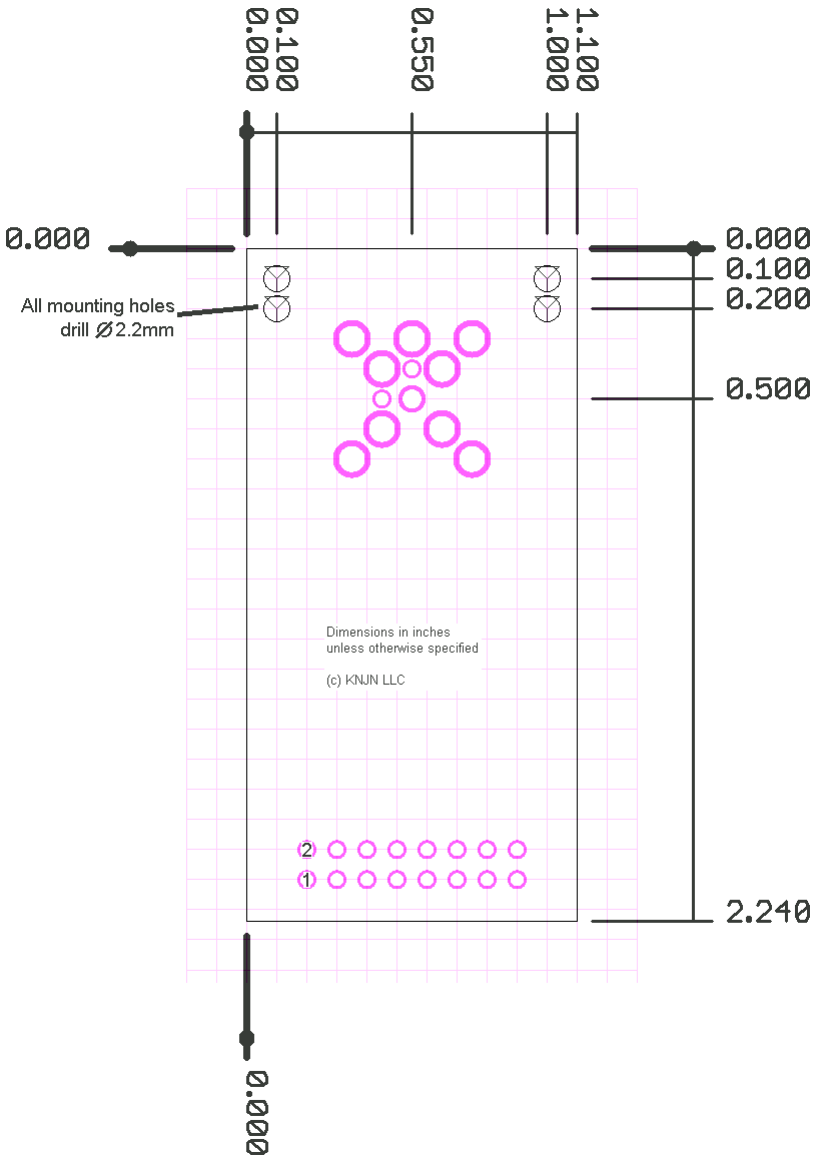
15.3 Dragon with Ethernet connectors

When using Flashy or FlashyD at 100MHz on Dragon, you have the option to disconnect the Ethernet connectors from the FPGA by removing a few resistors from the Dragon board.

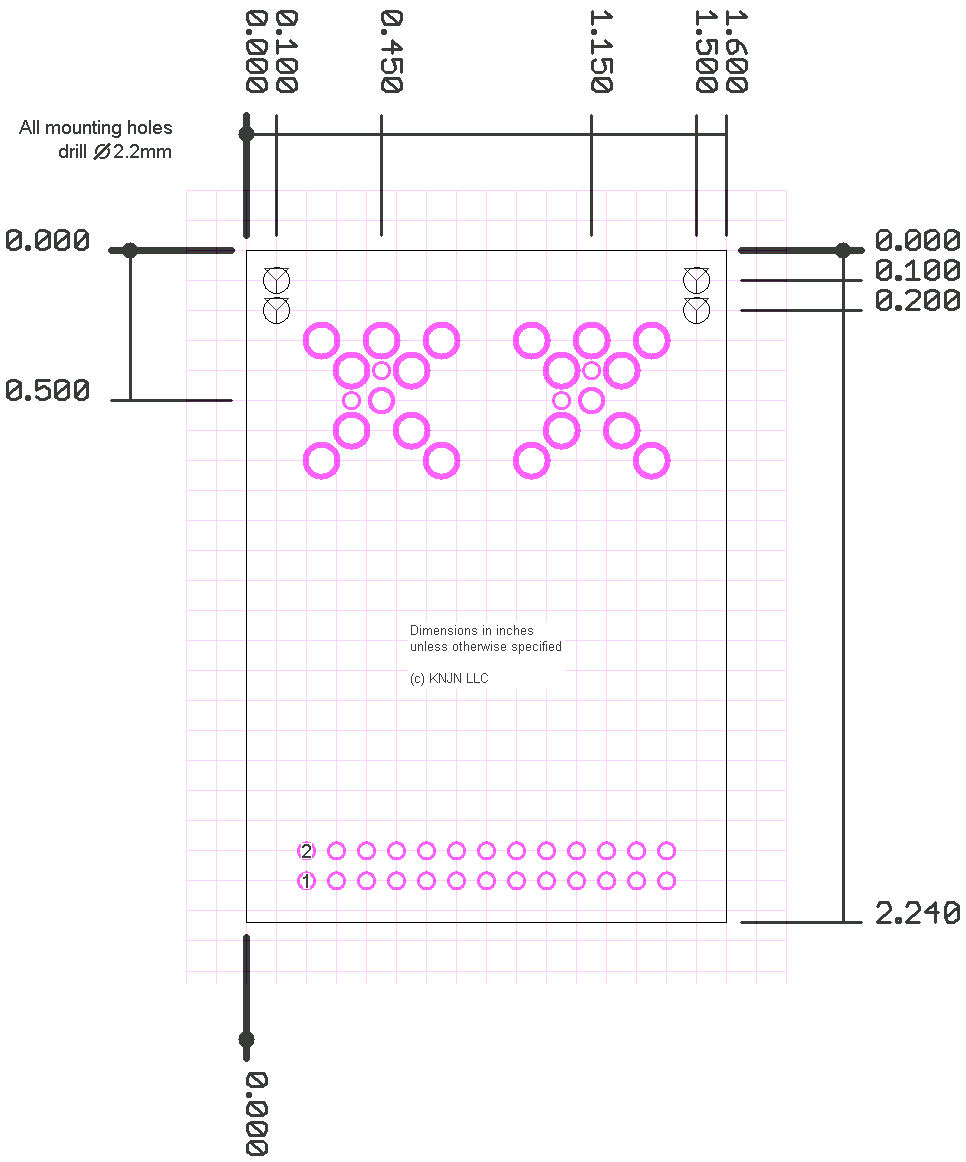
The Ethernet connectors share 6 traces with the FlashyD connectors. Six resistors are connected on these traces. These resistors are placed along the FlashyD connector. Removing the resistors avoids the trace stubs that go to the Ethernet connectors. That allows for better signal integrity. If you experience occasional spikes on the oscilloscope display, you may want to remove these 6 resistors.

16 Mechanical drawings

16.1 Flashy



16.2 FlashyD


0.000
0.100
0.450
1.150
1.500
1.600
All mounting holes
drill ∅2.2mm
0.000
0.500
0.000
0.100
0.200
Dimensions in inches
unless otherwise specified
(c) KNJN LLC
2
1
2.240
0.000