LinuxFocus article number 300
http://linuxfocus.org

A 1 Bit Data Scope

by Bob Smith
<bob(Q)linuxtoys.org>

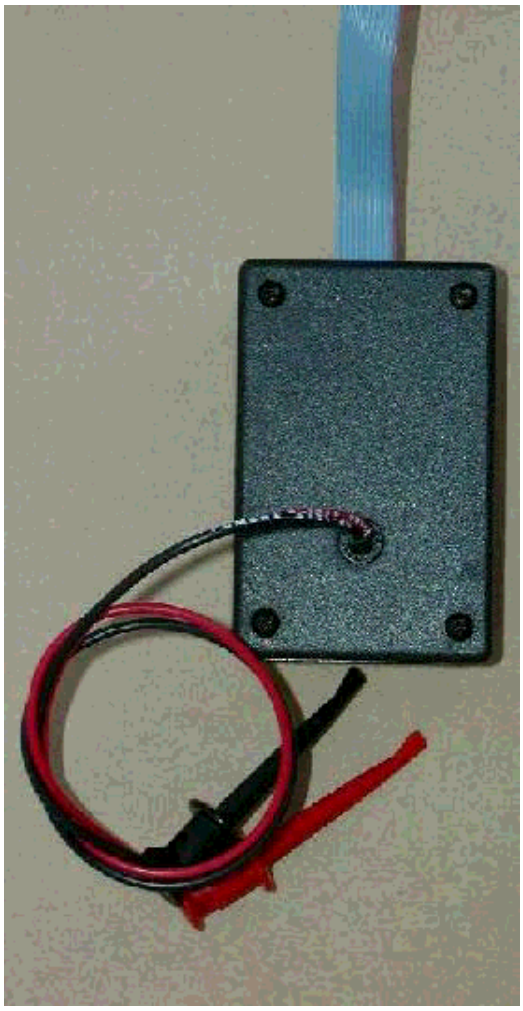
About the author:

Bob is an electronics hobbyist and Linux programmer. You can find his latest project at www.linuxappliancedesign.com and his homepage at www.linuxtoys.org.

Abstract:

The 1 Bit Data Scope captures one bit of data at a rate of 46080 (or 92160) samples per second and encodes the data into a stream of RS-232 characters which can be read at 57.6 (or 115.2) kilobaud. The Data Scope is a fairly simple construction project which uses six low cost IC’s.

Introduction



This article describes the design, construction, and use of the 1 Bit Data Scope. A program to capture and display the data stream is given.

Design and Theory of Operation

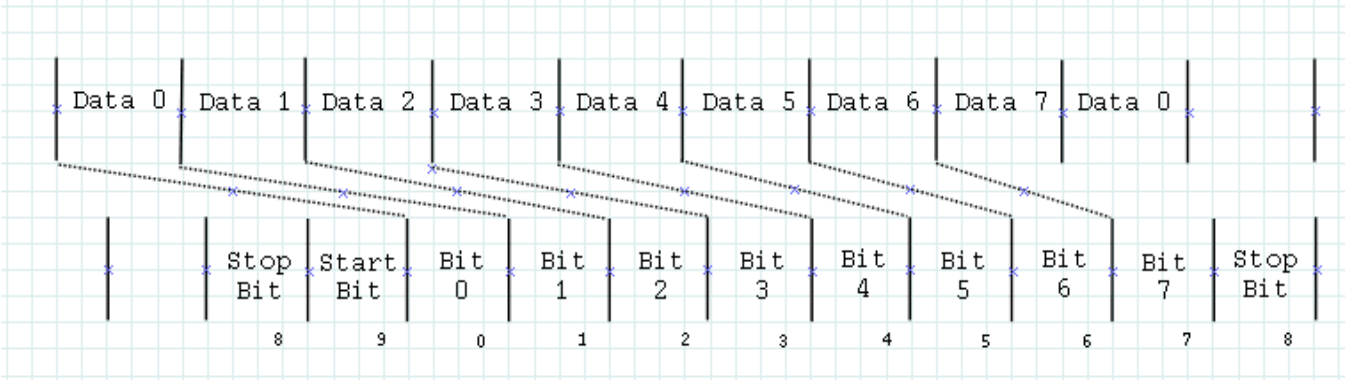
Design Goals

- Minimum sample rate to be in excess of 8000 samples per second
- Output data to a PC serial port
- Continuously sample data without missing a single bit
- Be parasitically powered by the serial port

Design Overview

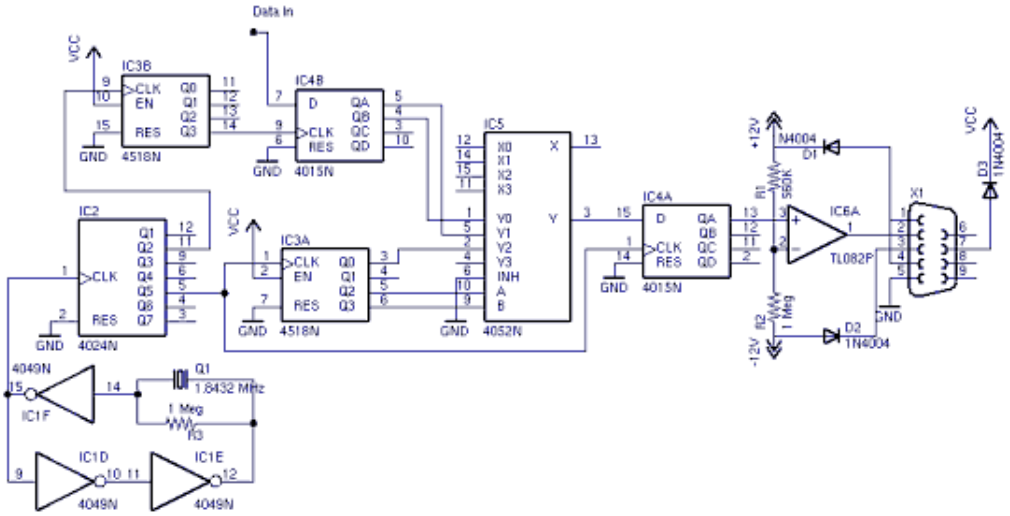
The goal is to sample an input at a regular frequency and output the data framed in a serial character with one start bit, eight data bits, and one stop bit. The overhead of the start and stop bits implies that the data sample rate must be eight-tenths of the serial baud rate. For a 57.6 kbaud serial port the data rate is (8 * 5.76 k) or 46080 samples per second.

The problem is that we need to store the input samples while we are sending the start and stop bits. The following diagram illustrates the problem.



During bits 0, 1, 2, and 3, we want the data delayed by two data clocks, so we select Q(B) from the 4015 shift register. During bits 4, 5, 6, and 7, we want the data delayed by one data clock, so we select Q(A) from the shift register. During the stop and start bits (bits 8 and 9) we want a zero and one, so we select Q(0) of the bit counter.

Schematic of the 1 Bit Data Scope



The schematic capture was done with a very nice package called "EAGLE" available from Cadsoft. The schematic file for the above circuit is available as 1bitla.sch.gz.

The Clock Circuit

The clock circuit has two outputs: one at the baud clock and one at eight-tenths of the baud clock. The clock circuit uses three inverters from the 4049, the 7-stage binary counter in the 4024, and a decade counter from one-half of the 4518. The oscillator runs at 1.8432 MHz and Q(2) of the 4024 has a square wave at 460.8 kHz. The Q(3) output of the divide by ten counter, 4518, is 46.08 kHz. The Q(5) output of the 7-stage ripple counter has a square wave at 57.6 kHz.

To run the circuit at 115.2 kHz, use the Q(1) and Q(4) outputs of the 4024 instead of the Q(2) and Q(5) outputs. Other baud rates are possible by using a different value for the crystal or by replacing the crystal oscillator with an RC oscillator.

You may find that the crystal oscillator draws less power than an RC circuit at the same frequency. To conserve power be sure to ground the unused inputs of the 4049 hex inverter.

Input Buffering

There are two easy ways to add input buffering to the above circuit. You can connect the data input to the input of one of the unused inverters in the the 4049. This really does not buy too much though since the input presents one CMOS load and still has CMOS thresholds (which is at about three volts in this circuit).

A better input buffer might be to use the second op-amp in the TL082. You can use a resistor divider attached to one op-amp input and use the other input for the data in. This lets you control the threshold very accurately and presents a very high impedance to the device under test.

CMOS to RS-232 Converter

One-half of the TL082 op-amp provides CMOS to RS-232 conversion. The resistor divider R1, R2 biases the negative input to about 3 volts. As the output of the 4015 register goes above and below 3 volts, the output of the op-amp goes to positive and negative supply voltages. Note that the sample data on the RS-232 line is inverted, and the software reading the data must take this inversion into account.

Power Supply

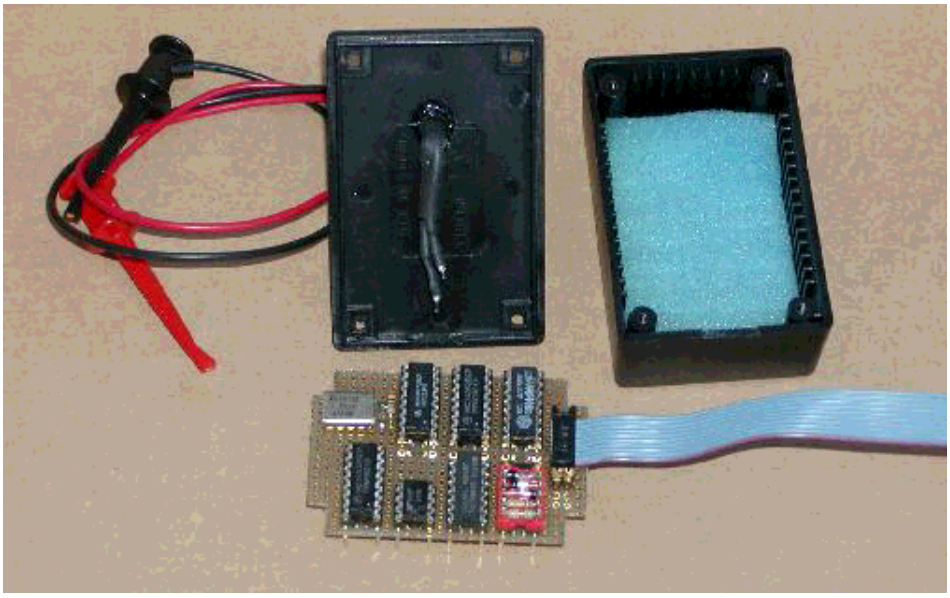
Power for the circuit is taken from the RS-232 lines from the computer. Tx from the computer supplies the negative voltage for the RS-232 converter (the TL082P) and DTR provides the positive voltage. In the prototype the diodes D1 and D2 were not really needed. You might try the circuit without them.

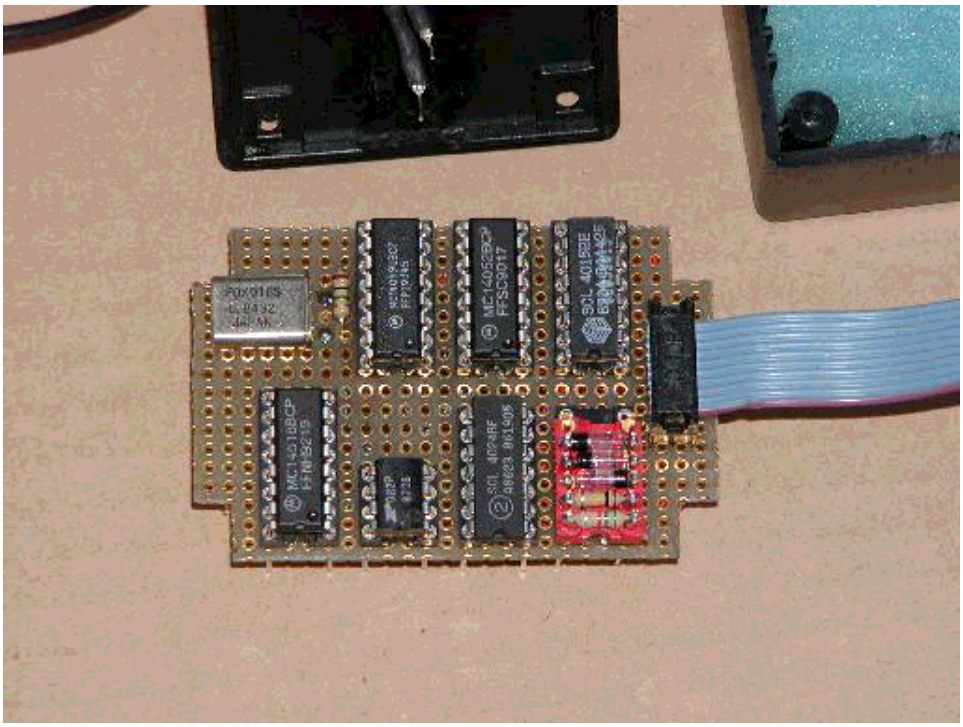
RTS provide power (Vcc) for the rest of circuit. In the prototype Vcc was about 7 volts. Your voltage for Vcc may vary depending on the computer's RS-232 interface. The software driving the Data Scope must set both DTR and RTS high.

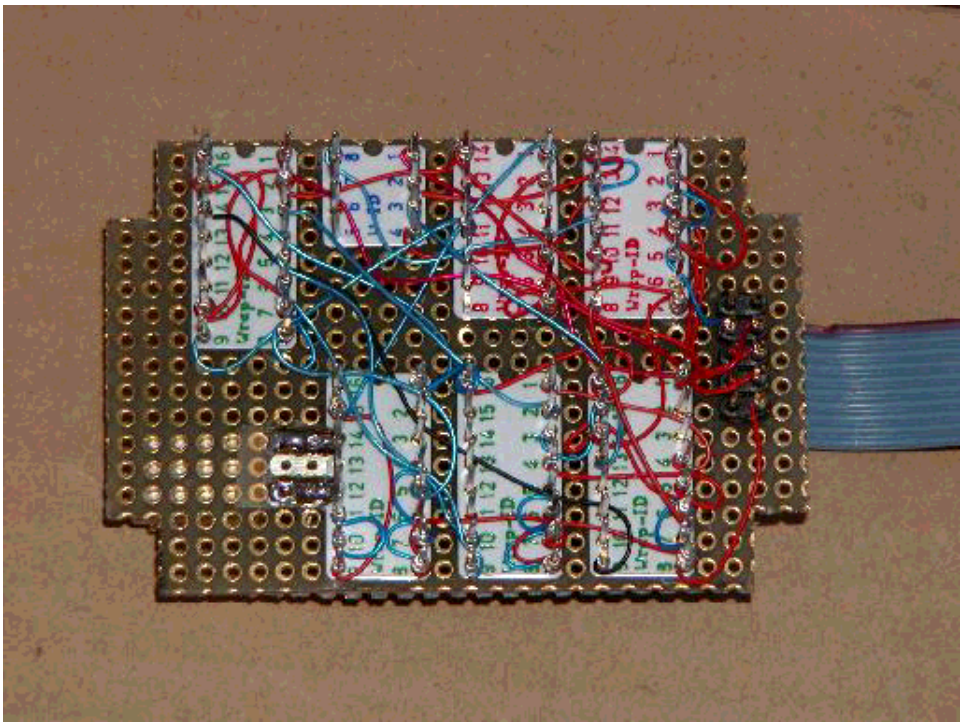
Construction

The frequencies in the Data Scope are relatively low and almost any construction technique will work. Wire wrap was used in the prototype with pretty good results. You may want to build and route the power supply first. The IC's have Vcc on pin 14/16 as expected except for the 4049 hex inverter which has Vcc on pin 1.

Here are some photos to show you one approach to the construction.







Simple Data Capture Program

The simple program here (1bitla.c) captures the serial data and can output the data in three formats. The first format is the raw data, displayed one byte per line with a two character hex display. Note that the program inverts the data before display, and that the data is sent MSB first. Raw output is invoked with the -r command line option. The second format is a value (0 or 1) followed by a count of how many one bit samples had that value. The value/count output is invoked with the -c option. The third output format is a value (0 or 1) followed by the number of seconds the input had that value. The resolution at 46080 kHz is 21.7 microseconds. The value/time output is invoked with the -t option.

The program is invoked as
`1bitla [option] serial_port`
where option can be -r, -c, or -t. The default is value/count (-c).

The design of the program is fairly simple. We process the command line options, open the serial port, and loop forever reading and displaying bytes. Since we are not doing any background processing we use a blocking read.

You may be satisfied with just displaying the samples to the console or redirecting them into a file for later processing, or you might want to perform a true "logic analysis" on the data by building a state machine to process the samples. Example state machines might include decoding commands sent from an infrared remote control, or decoding the position of a radio controlled pulse width modulated output.

Sample Output

-r	-c	-t
03	0, 0	0, 0.000000
e3	1, 5	1, 0.000065
ff	0, 3	0, 0.000065
03	1, 10	1, 0.000217
e3	0, 6	0, 0.000130
ff	1, 5	1, 0.000109
03	0, 3	0, 0.000065
e3	1, 10	1, 0.000217
ff	0, 6	0, 0.000130
03	1, 5	1, 0.000109
e3	0, 3	0, 0.000065
ff	1, 10	1, 0.000217
03	0, 6	0, 0.000130
e3	1, 5	1, 0.000109
ff	0, 3	0, 0.000065
03	1, 10	1, 0.000217
e3	0, 6	0, 0.000130
ff	1, 5	1, 0.000109
03	0, 3	0, 0.000065
e3	1, 10	1, 0.000217
ff	0, 6	0, 0.000130
03	1, 5	1, 0.000109
e3	0, 3	0, 0.000065
ff	1, 10	1, 0.000217
03	0, 6	0, 0.000130
e3	1, 5	1, 0.000109
ff	0, 3	0, 0.000065
03	1, 10	1, 0.000217
e3		
ff		
03		
e3		
ff		
03		
e3		

References

- Download page for this article

- http://www.linuxtoys.org/, this and other hardware project for Linux

---

2005-01-14, generated by lfparser_pdf version 2.51