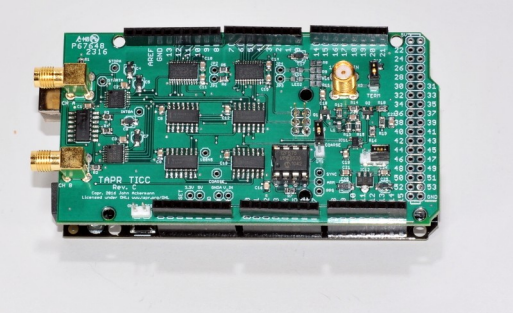
TAPR TICC Timestamping Counter Operation Manual

Revised: 23 November 2016



Introduction

The TAPR TICC is a two-channel timestamping counter ("TSC") implemented as a "shield" daughterboard for an Arduino Mega 2560 controller. It can perform more than 100 measurements per second (both channels) with resolution of about 60 picoseconds and RMS jitter of less than 100 picoseconds – that yields a one-second noise Allan Deviation of about $7x10^{-11}$ with a slope of -1 at longer measurement intervals.

A timestamping counter is a bit like the time clock at a business where each employee "punches in" and the time is recorded. The output from a TSC is a record of the arrival time (in seconds since start-up) of each input event, measured against the counter's reference clock. For example, a series of pulse-per-second events might look like this:

```
104.897999794440
105.897999794492
106.897999794549
107.897999794551
108.897999794553
109.897999794552
110.897999794667
```

Note that this data increments by one second for each reading; that would be expected for a pulse-per-second input (if the input were at a 10 PPS rate, each reading would increment by 0.1 second). But also note that the interval isn't *exactly* one second – no two clocks have exactly the same rate, and they all have some amount of "noise" in their readings. From timestamp information one can determine frequency offset (difference between measured and nominal event rate) and stability ("noise" from reading to reading). Thus, a TSC can be used to characterize clock performance.

A single TSC channel compares a low repetition rate source such as a PPS signal against a reference oscillator with an "RF" output (in the TICC's case, 10 MHz). To measure the time interval between two PPS signals – for example the output of a clock and a GPS timing receiver – a two-channel counter can be used. Each channel is referenced to the same time scale, and generates a timestamp each time it sees an event on its input. By subtracting one reading of the two-channel pair from the other, the time between the two events can be determined; this is the equivalent of the "time interval" mode offered by traditional counters. Time interval data can be used to determine the frequency difference and other information related to the two input signals.

When measuring time interval, the reference clock serves only as a transfer standard and within reason, its quality does not impact the results. Thus an inexpensive reference can be used to measure a pair of very high quality devices.

A TSC can also derive other information from timestamp data, such as period (current timestamp minus last timestamp), ratio (number of pulses on channel A compared to number on channel B), etc.

The TICC has unusually good single-shot resolution of about 60 picoseconds. This is comparable to the best time interval counters commercially available today. High resolution allows more meaningful measurement results in a shorter time. For example, the TICC's noise (measured with the Allan Deviation, or ADEV, statistic) is below is well below $1x10^{-10}$ in one second. Because of its unique design, the TICC requires no calibration.[1]

The TICC outputs data in ASCII serial format via USB. The output files can be read by standard analysis software such as W. J. Riley's Stable32, and John Miles' TimeLab software can directly interface with the TICC.

1 There may be opportunities to improve performance via individual calibration and tweaking of some program variables, but this is not necessary for normal operation.

Operation

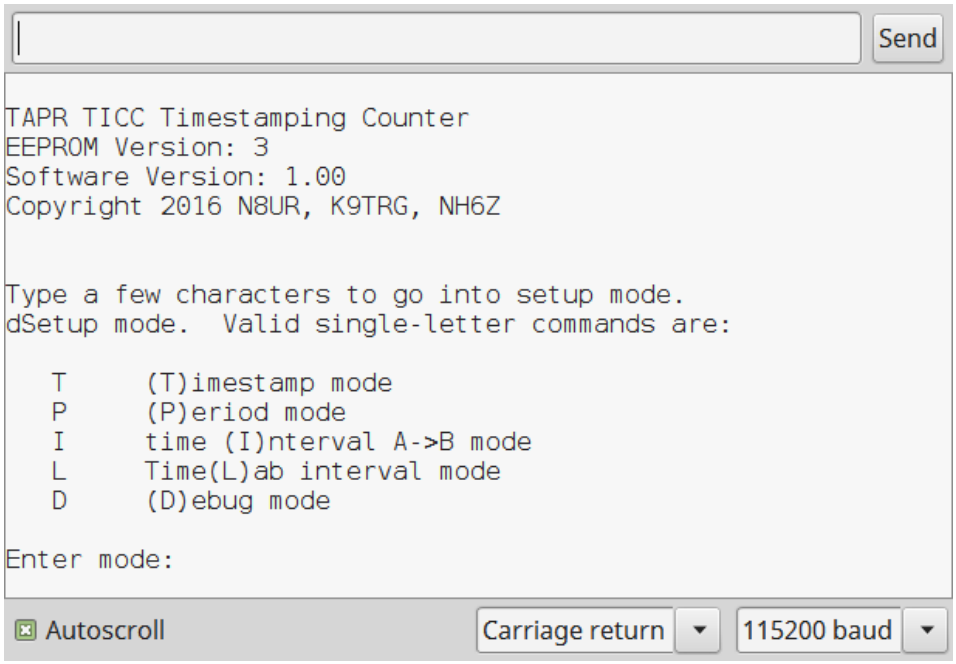
The TICC circuit board is a "shield" that mounts to an Arduino Mega 2560 processor. The Arduino is loaded with the TICC software via its USB connection. The software is available at https://github.com/TAPR/TICC. TICC systems provided by TAPR will have the board and processor mated, and software already loaded.

Connect a 10 MHz reference source to the vertical SMA receptacle on the board, which feeds a sine-to-square wave converter that accepts signals from -10 to +13 dBm. A jumper allows optional 50 ohm termination.

The channel "A" and "B" SMA connectors accept digital signal levels of up to 5 volts without damage. The trigger level is about 1.7 volts. The input impedance is 1 megohm. By default, the TICC triggers on the rising edge of an input signal. Future configuration options will allow falling-edge trigger for either channel, but the TDC7200 datasheet recommends against this when best performance is desired.

Communication with the TICC occurs over USB with serial port emulation. The port parameters are 115200, 8N1. All data is 7-bit ASCII.

At startup you will see a startup menu similar to the one below if you press a key within 5 seconds. If you do not press a key, the TICC will start up with the last saved configuration.



Timestamp outputs the timestamp of each event received on either channel A or B as it is received. Each measurement includes "chA" or "chB" to identify the channel. One or both channels may provide input, and inputs may be turned on or off at any time.

Period outputs the difference between the current reading for a channel and the last reading for that channel. Each measurement includes "chA" or "chB" to identify the channel. One or both channels may provide input, and inputs may be turned on or off at any time.

Time Interval outputs the difference of a pair of measurements. When a new event occurs on each of channels A and B, the TICC subtracts the channel A timestamp from the channel B timestamp and outputs the difference. No output occurs until a pair of readings have occurred. Both channels must be fed with events at

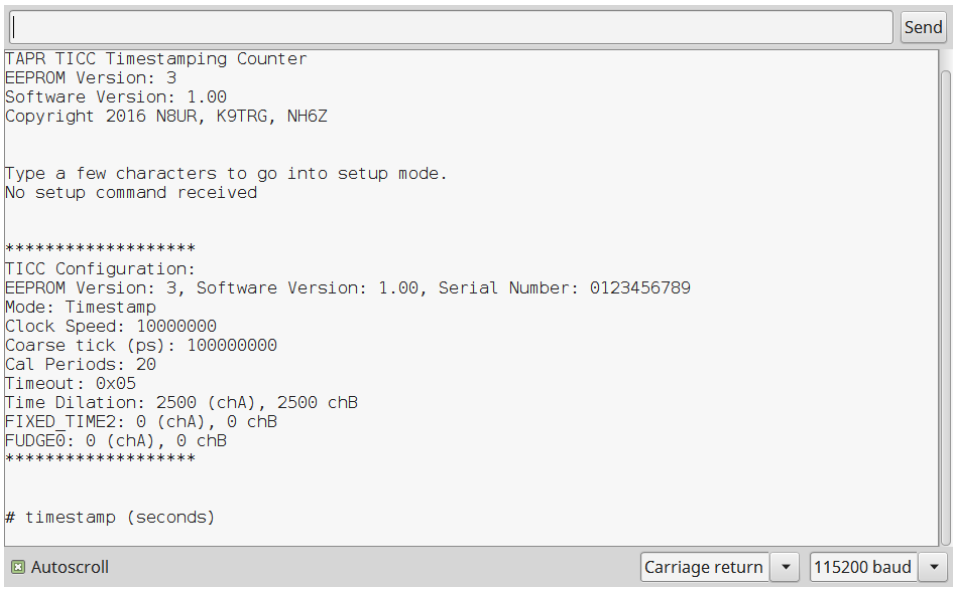
the same nominal rate.

TimeLab outputs three measurements for each pair of readings: channel A timestamp, channel B timestamp, and a pseudo-timestamp labled as "channel C" that consists of the (chB – chA) time interval added to the integer part of the channel B timestamp. This mode works with the multi-channel input capability of the TimeLab software to allow 3-corner hat measurements. This mode has not yet been thoroughly tested.

Debug outputs the raw data output from the TDC7200 chip, intermediate results, and the final timestamp calculation. The output fields are: time1Result, time2Result, Clock1Result, Cal1Result, Cal2Result, time-of-flight, PICcount, timestamp, and channel identifier.

All data fields output by the TICC are units of seconds with 12 decimal places. The least significant digit is 1 picosecond.

After the configuration screen, you will see a screen that shows the configuration parameter settings:



If the results you see look strange compared to the above, it's possible that the configuration values have not been written to the Arduino EEPROM; this should happen automatically.

Once the screen above displays, the TICC will generate no further output until it has measurement results to report.

Circuit Description

A limitation of digital counters is that their resolution is tied to the clock speed. If there is only one clock tick every millisecond, you cannot measure with greater than one millisecond resolution. A clock rate of 1 GHz, which is challenging to achieve, provides a resolution of one nanosecond.

In order to obtain higher resolution, it is necessary to interpolate between clock cycles. Traditionally, this has required analog circuits using methods such as measuring the decreasing voltage over time across the terminals of a capacitor. The best of these schemes can yield resolution and jitter less than 100 picoseconds, but they are complex and require periodic calibration.

The TICC uses a different method to measure sub-clock-cycle times, thanks to the Texas Instruments TDC7200 time-to-data converter chip. The TDC7200 is at the core of the TICC's design, but a significant amount of additional logic is required to create a complete timestamping counter.

High Level Design

The TICC is a clock that measures when external events (logic pulses on the TICC inputs) occur. It does this through a combination of hardware on the TICC shield, and software on the Arduino. It's difficult to look at either the hardware schematic or the software source code in isolation to gain an understanding of the system. This high-level design description should help.

Central to the design is a software counter implemented in the Arduino that counts the number of 100us intervals since system startup. The counter is a variable called `PICcount`, and it increments via hardware interrupts each time a 10 kHz clock (`COARSE_CLOCK`) on the TICC board ticks. `PICcount` is therefore a timescale based on the number of 100 us ticks since the system started.[2] Note that "time" referred to here is not related to an outside timescale like UTC; it starts from zero each time the system initializes.

The goal of the TICC is to provide data with resolution measured in picoseconds, and by itself the 100us tick of the `COARSE_CLOCK` does not come close to meeting that requirement. The TDC7200 chip can measure time intervals with <60 picosecond resolution, so it is used to measure the time of an event within the window between two `COARSE_CLOCK` ticks.

When an event appears at a TICC input, the associated TDC7200 chip starts its measurement.[3]

Once the TDC chip has started timing, the next `COARSE_CLOCK` tick to arrive[4] stops the measurement, and the TDC then calculates the elapsed time, called the "time of flight" or "`TOF`," and sends that value to the Arduino via the SPI communication bus. The same `COARSE_CLOCK` tick also raises an Arduino hardware interrupt that causes the software to copy the current value of `PICcount` into another variable called `PICstop`.[5] The value in `PICstop` is always a timestamp *after* the event occurred.

The `TOF` measurement tells us, with high resolution, the time from the event until the timestamp captured in `PICstop`. So *subtracting* `TOF` from `PICstop` gives the event's actual timestamp with the full resolution of the TDC7200. The TICC software calculates the timestamp to one picosecond resolution and either directly outputs this value on the serial port, or uses it as part of a time interval or nother calculation.

The TICC has two input channels that operate independently, but work with the same time scale. Thus

---

2 `PICcount` is a 64 bit variable. Even at a 100 us clock rate, it would take millions of years to overflow.

3 See the next section for more details on how the TDC7200 does this.

4 This is not quite true. The TDC chip requires a minimum time period between its "START" and "STOP" inputs. A circuit described below called the "STOP GATE" ensures that the minimum time period is met and could result in the second, rather than the first, COARSE_CLOCK tick be the one that is used.

5 PICstop is stored separately for each of the two channels.

measurements on each channel can be compared with one another. Multiple TICC boards can be synchronized to allow comparisons of 4, 6, 8, or more channels. Multiple-unit use is described in an Appendix.

TDC7200 Operation

Texas Instruments designed the TDC7200 to measure the flow rate of fluids using ultrasonic transducers. In normal use, it starts timing when a "ping" is sent and stops when the transducer hears one or more echoes – the resulting time of flight value can be used to calculate the speed of the fluid flow. However, the chip can be used for other purposes. It is essentially a stopwatch with extremely high resolution. Its START and STOP pins serve as signal inputs.

Like the TICC at a higher level, the TDC7200 has two timing circuits, one (relatively) coarse and the other (very) fine, used to measure the time between a pulse arriving on the chip's START pin and another arriving on the STOP pin. The coarse timer is a conventional digital counter using a 10 MHz clock to provide 100ns resolution, while the fine timer is a ring oscillator with 63 inverter stages providing a period of about 57 ps[6] that interpolates between clock ticks. The 10 MHz clock is the same one used to derive the COARSE_CLOCK tick, so all timing on the TICC board is synchronous.

A ring oscillator[7] is a chain of an odd number of digital logic inverters hooked input-to-output – sometimes described as "a snake eating its own tail." The feedback from the end of the chain to the beginning results in a free-running oscillator with a period based on the propagation delay of the gates. The TDC7200 fine timer counts the number of times the around the ring.

While the ring oscillator is very fast, it is not stabilized and its speed varies with temperature and other factors. One of the really clever features in the TDC7200 design is the calibration cycle that occurs at the end of every measurement where the chip counts the number of ring oscillator periods over a a number of 100 ns ticks of the external 10 MHz reference. From that data the Arduino software can calculate the actual period of the ring counter at the time of the measurement. This provides temperature compensation among other things.

When the TDC7200 sees a START signal, the ring oscillator cycles until the next 100 ns clock edge. When a STOP signal arrives, the ring counter starts again and continues again until the next edge of the clock. The illustration below (from the TDC7200 datasheet) shows how this works.

6 TI states that the nominal resolution is 55 picoseconds, but our measurements consistently show 57+ ps.
7 See https://en.wikipedia.org/wiki/Ring_oscillator

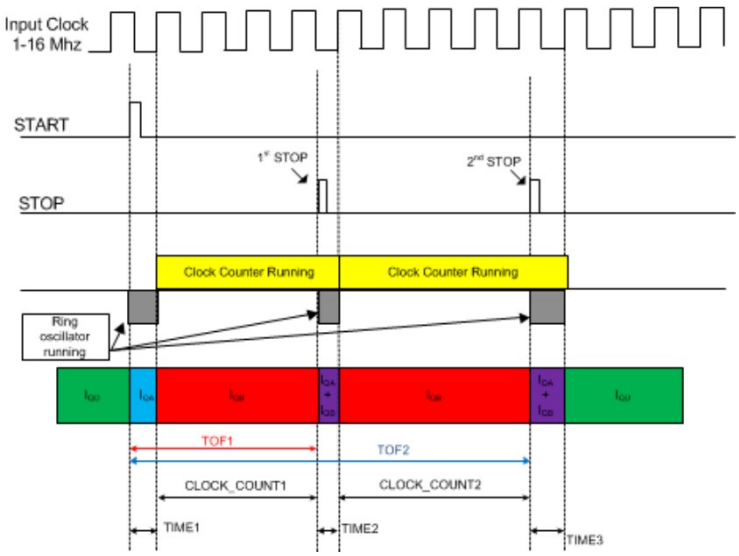


Figure 18. Measurement Mode 2

The elapsed time (what TI calls "time of flight" or "TOF") is thus TIME1 + CLOCK_COUNT1 – TIME2. (In some applications, multiple STOP signals may be received after a single START signal, so the illustration shows additional measurements. The TICC uses only the first STOP pulse.)

The TDC7200 has two limitations that prevent it from being a useful standalone time interval counter: first, the time from START to STOP must be greater than 12 nanoseconds; and second, the maximum time it can measure is about 6 milliseconds. Thus, it won't work for general-purpose use.

The first limitations is overcome by the STOP GATE circuit described below, while the `COARSE_CLOCK` on the TICC and `PICcount` timer on the Arduino address the second.

TICC Timestamp Logic

The TICC adds circuitry around the TDC7200, together with software in the Arduino processor to which the TICC is mounted, that creates a fully functional timestamping counter.

A PIC chip loaded with Tom Van Baak's frequency PD15 divider firmware generates 100 us pulses from the same 10 MHz reference that drives the TDC7200. In addition to triggering an interrupt on the Arduino, the 100 us `COARSE_CLOCK` signal provides the STOP signal to the TDC7200 through a logic block I call the Stop Gate that consists of flip-flops and a shift register. The Stop Gate ensures that the TDC7200's 12 ns minimum time requirement is met by passing only a `COARSE_CLOCK` tick that meets the timing requirement.

The rising edge of each pulse that arrives from the device under test ("DUT") triggers the START pin of the associated TDC7200 chip, and also arms the Stop Gate. The Stop Gate is clocked by the 100ns system clock and waits for three system clocks (300ns) before allowing a `COARSE_CLOCK` tick to pass. Once that time has passed, the next `COARSE_CLOCK` will be routed to the TDC7200 STOP pin and the Arduino hardware interrupt pin. This ensures that the TDC7200 minimum START-to-STOP time is met. As a result, the TOF will range from a minimum of 300 ns (COARSE_CLOCK arrives just as the gate opens) to 100.299...us (COARSE_CLOCK arrived just under 300 ns after, so wait for one full cycle).

The following diagram shows the overall functional diagram of one TICC hardware channel.

TICC FUNCTIONAL DIAGRAM
6 March 2016

One of Two TSC Channels