Christine Ku

TDC Documentation

Background

Project Overview

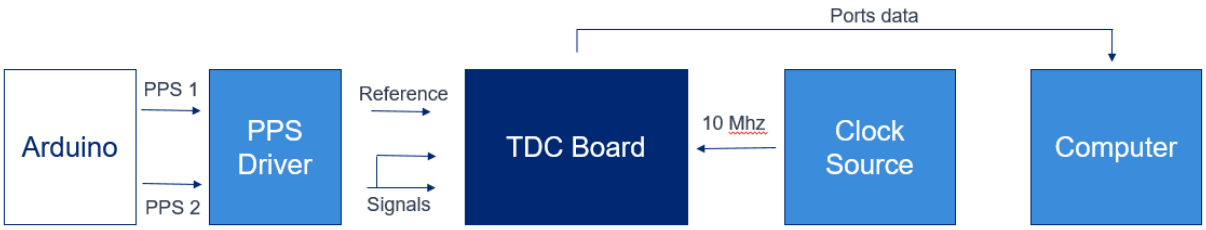
The Integrated Positioning, Navigation, and Timing (iPNT) system was developed for the [Redacted] as a situational awareness tool to help identify and help support the fleet in GPS-challenged environments. The system was initially developed as a Temporary Alteration and installed on the [Redacted] for one deployment cycle in [Redacted]. Since then, [Redacted] has worked to integrate the system as a [Redacted] element of the [Redacted] organic navigation suite. The latest TI-20 iteration of the iPNT system will be fielded as a part of [Redacted] and will be installed on [Redacted] platforms.

The TI20 system builds off of previous iterations of the system developed for [Redacted] and [Redacted] platforms as a part of the [Redacted]. The TI20 system aims to miniaturize COTS hardware selected as part of the TI18 systems into custom [Redacted] designed board solutions. This allows for the integration of additional hardware and sensors into the same footprint. The system also has some capabilities ported to a VPX card, which allows for increased modularity and an open architecture.

The custom timing board specifically is designed to replace two COTs components that were located within the TI18 chassis: a [Redacted] reference clock and a [Redacted] Time Interval Counter. The TDC ([Redacted]) circuit specifically reduces the footprint of the COTS, and provides us with additional channels to compare 1PPS references from various ships sensors along with additional organic sensors internal to our chassis.

Remote Setup

The picture above shows the remote setup that was used when adding functionality to the TDC (This image has been redacted.), and the diagram below is a more clear depiction of how the different components are connected.



Purpose

The purpose of the TDC ([Redacted]) is to monitor the integrity of signals from various timing sources (e.g. GPS, LORAN), compare the references to one another, and potentially calibrate them.

Functionality

Polling Signals

Originally, the board would take in signals from a few sources and could recognize them if they are there on the board's startup. However, if you disconnect and reconnect one of the signals or just add a completely new signal, the board would not be able to detect it. Now, the board can detect new signals regardless of when it is connected. Signals can also be disconnected and reconnected at any point.

User Configuration

Upon startup, the serial terminal will output the data in normal mode (printing the 12 ports in nanoseconds). If the user presses "r", they can access a menu to change several features:

m – Normal or verbose mode

- Normal mode outputs the values in nanoseconds
- Verbose mode outputs the raw data values read from the indices

p – Port selection

- Allows the user to read from 1-12 ports on the TDC
- Values 10 and 11 are selected using hex "a" or "b"
- To select all 12 ports, leave as default.
- Standard order for individual testing:
    - Start at TDC-0 B, skipping the A ports of each chip
    - Select port alphabetically and chip numerically.
    - i.e. after TDC-0 B, test TDC-0 C and TDC-0 D, then TDC-1 B, TDC-1 C, etc.

d – Default settings

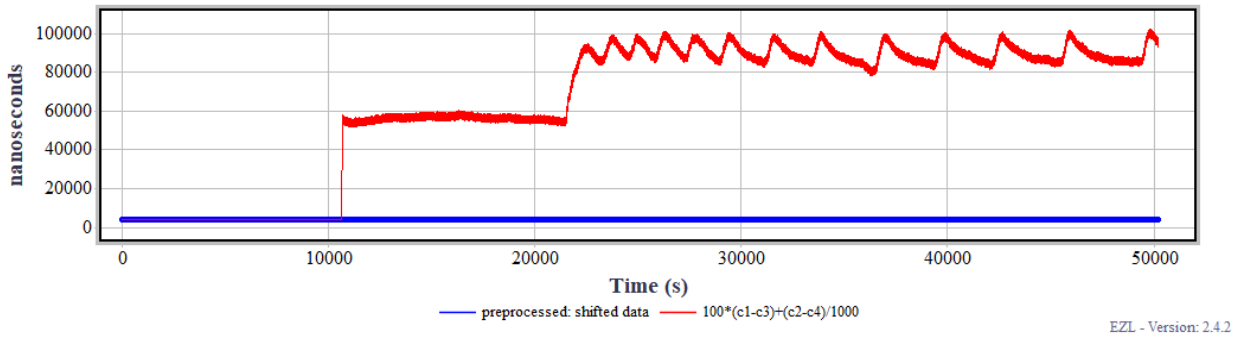
- Normal mode
- Reads 12 pins

s – Start measurements

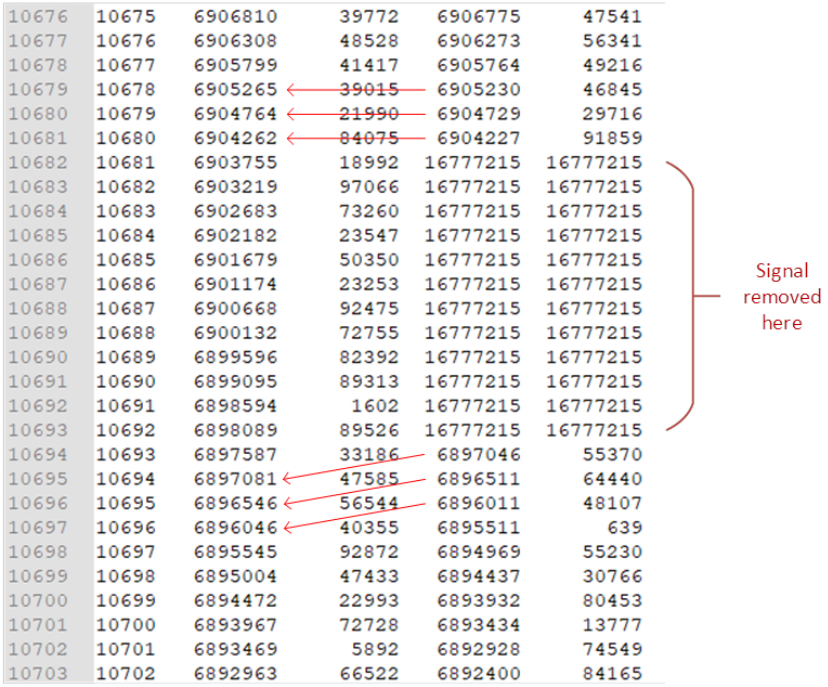
- After the user is done with the configuration, this command is used to resume taking measurements.

Troubleshooting

While testing the continuous polling feature, some unexpected behavior appeared. When a signal was disconnected and reconnected, the error would jump from 3500 (to 60000 when disconnected) to 100000 ns as shown in the diagram below. This issue seemed to have occurred more readily after the TDC board had been sending data for more than 3 hours, but it was still possible to recreate the behavior within a few seconds by disconnecting and reconnecting sporadically. It is also more difficult to replicate in the lab because the integrity of the PPS in the lab is higher than the Arduino sourced one.



Upon further investigation, it seemed like the data became misaligned as shown in the image below.



This problem seemed to have stemmed from some FIFO buffers being filled completely while others still had empty space. When a PPS signal is pulled off, that channel's FIFO is cleared, but in order to restart it, 2 STOPs are needed. The other channels had already executed 2 STOPs at the beginning of the loop, so this discrepancy in time makes the data lag.

To prevent this from happening, the COMMON_FIFO_READ configuration was set to 0 so that all four channels are completely independent from one another. If there are any pending measurements/empty spaces in the FIFOs, the INTERRUPT pin will be held low (from the data sheet: "interrupt gets high when all FIFO are empty"). In tdc_spi.cpp, the code has been changed to make sure the measurements are all read through the buffers as long as there is at least one FIFO with valid data.

Results

Now, the problem of disconnecting and reconnecting does not appear. (The results and graphs have been redacted.) It was tested in lab and at home over several hours and multiple reconnections as shown below:

Future Features/Investigation

There are always more features that could be added or updated. For example, the menu could be fleshed out to be a more user friendly and robust user configuration tool.

The new board design (as of [Redacted]) has a few hardware changes including an external interrupt that would need to be added into the configuration file. The code would need to be adapted for a different interrupt source.

The placement of the interrupt will be something to think about as well. If the interrupt occurs somewhere between the reference pulse on Port A and a pulse on any other port, there is potential misalignment of measurements.

In this drawing above, the reported time difference between Port A and Port B will be 500. (This drawing has been redacted.) The difference between A and D will be 600. The difference between Port A and Port C *should* be 1000, but because the interrupt occurred, the buffers are cleared and the index is reset. Ports A, B, and D read out 0x00FFFFFF (overflow), but Port C reads out different values. This has not been an issue seen in the wild yet, but it is important to note in case it does in the future.