Christine Ku

TDC Documentation

Background

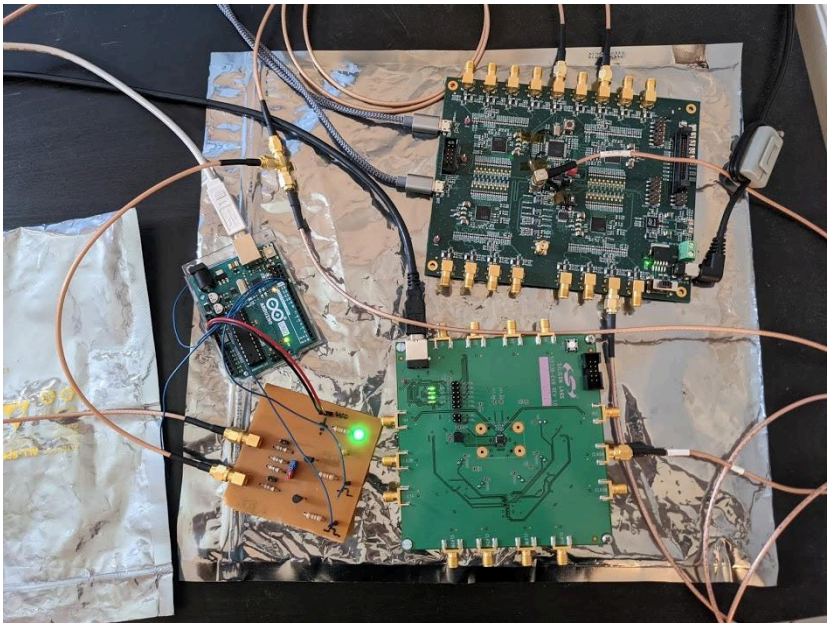
Project Overview

The Integrated Positioning, Navigation, and Timing (iPNT) system was developed for the Navy 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 USS Annapolis (SSN 760) for one deployment cycle in 2018/2019. Since then, JHU/APL has worked to integrate the system as a NAVWAR element of the submarine force's organic navigation suite. The latest TI-20 iteration of the iPNT system will be fielded as a part of Submarine Force Tech Insertion 2020 (TI20) and will be installed on Virginia Class SSN platforms.

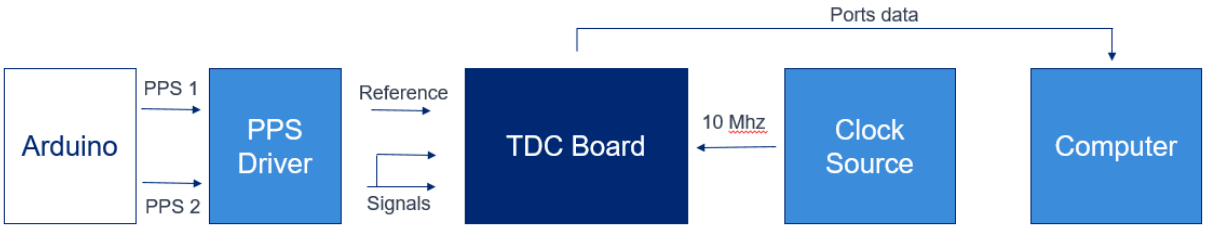
The TI20 system builds off of previous iterations of the system developed for SSN 688 and SSGN platforms as a part of the 2018 Tech Insertion. The TI20 system aims to miniaturize COTS hardware selected as part of the TI18 systems into custom APL designed board solutions. This allows for the integration of additional NAVWAR 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 Jackson Labs Rubidium reference clock and a 5 channel Time Interval Counter. The TDC circuit specifically reduces the footprint of the COTS TIC, 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 (WFH)



The picture above shows the home setup that was used when adding functionality to the TDC and the diagram below is a more clear depiction of how the different components are connected.



Purpose

The purpose of the TDC 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

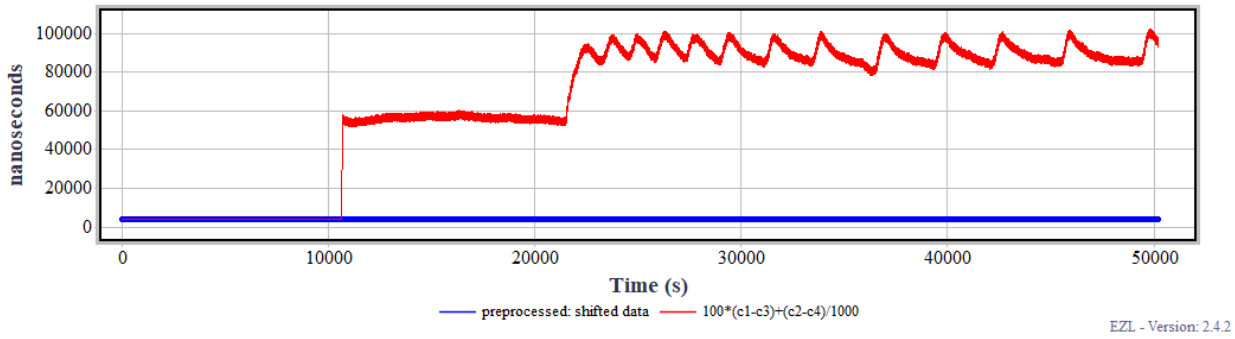
- Reads 12 pins

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 also more difficult to replicate in the lab, probably 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.

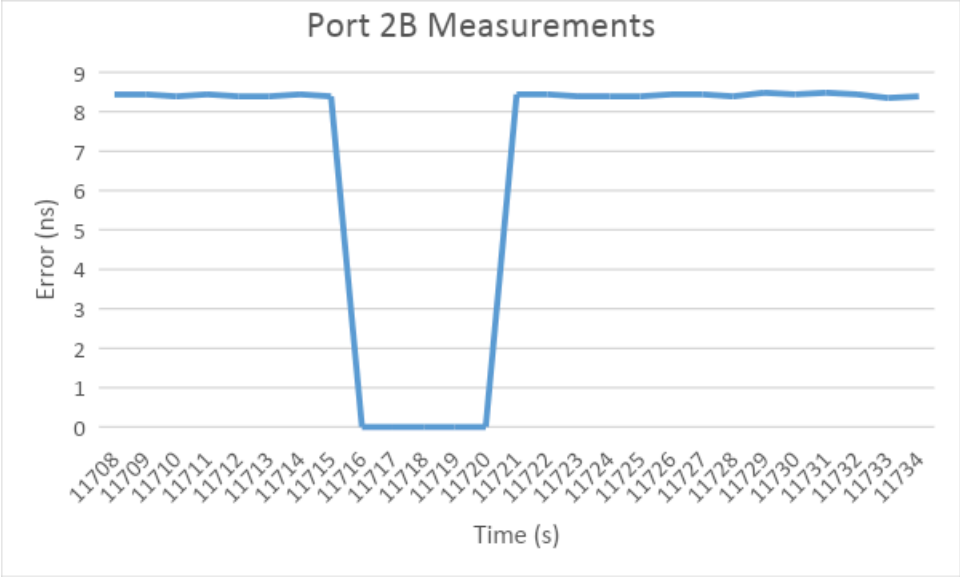
10676	10675	6906810	39772	6906775	47541
10677	10676	6906308	48528	6906273	56341
10678	10677	6905799	41417	6905764	49216
10679	10678	6905265	39015	6905230	46845
10680	10679	6904764	21990	6904729	29716
10681	10680	6904262	84075	6904227	91859
10682	10681	6903755	18992	16777215	16777215
10683	10682	6903219	97066	16777215	16777215
10684	10683	6902683	73260	16777215	16777215
10685	10684	6902182	23547	16777215	16777215
10686	10685	6901679	50350	16777215	16777215
10687	10686	6901174	23253	16777215	16777215
10688	10687	6900668	92475	16777215	16777215
10689	10688	6900132	72755	16777215	16777215
10690	10689	6899596	82392	16777215	16777215
10691	10690	6899095	89313	16777215	16777215
10692	10691	6898594	1602	16777215	16777215
10693	10692	6898089	89526	16777215	16777215
10694	10693	6897587	33186	6897046	55370
10695	10694	6897081	47585	6896511	64440
10696	10695	6896546	56544	6896011	48107
10697	10696	6896046	40355	6895511	639
10698	10697	6895545	92872	6894969	55230
10699	10698	6895004	47433	6894437	30766
10700	10699	6894472	22993	6893932	80453
10701	10700	6893967	72728	6893434	13777
10702	10701	6893469	5892	6892928	74549
10703	10702	6892963	66522	6892400	84165

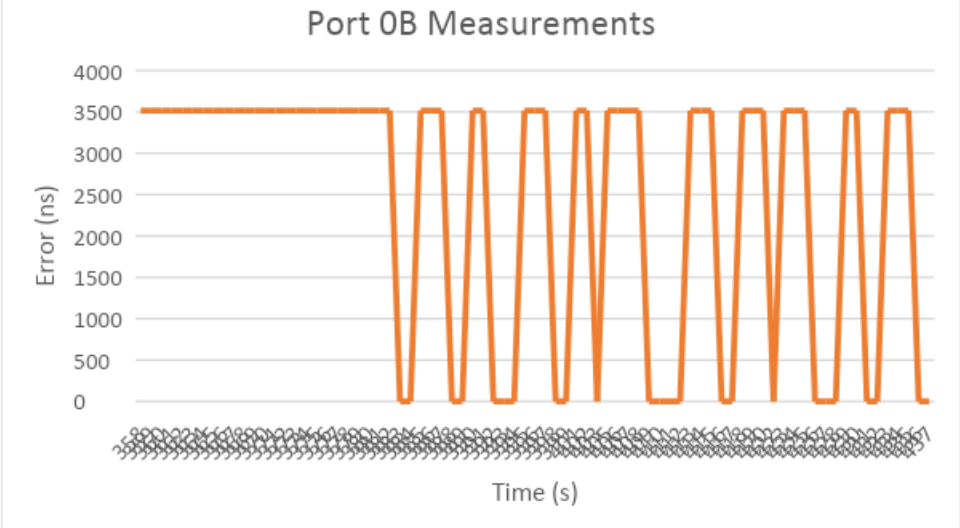
Signal removed here

This problem seemed to have stemmed from some FIFO buffers being filled completely while others still have empty space. When a PPS signal is pulled off, that channel's FIFO is cleared, but then in order to restart it, 2 STOPs are needed. The other channels have 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.

Now, the problem of disconnecting and reconnecting does not seem to appear. It was tested in lab and at home over several hours and multiple reconnections as shown below:



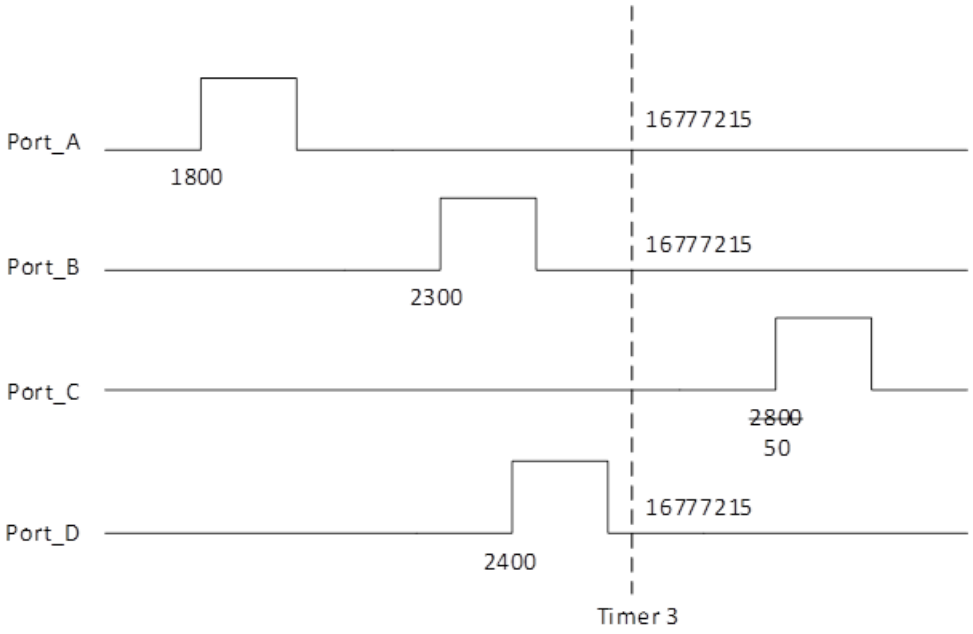


Future Features/Investigation

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 07/2021) has a few hardware changes including an external interrupt that would need to be added into the configuration file as well as adapting the code for a different interrupt source.

The placing 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 a potential misalignment of measurements.



In this drawing above the reported time difference between Port A and Port B will be 500. 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 is something else. This has not been an issue seen in the wild yet, but it is important to note in case it does in the future.