Design laboratory project	Time to Digital Converter	09.02.2026
Hubert Sierant, Lena Przybylska		WIET, AGH

1. Goal of the project

The goal of the project was to test and evaluate the ScioSense TDC-GPX2 Time-to-Digital Converter(TDC) board.

Initially, the device was operated using the manufacturer's software in order to verify the correct functionality.

In later stage, the board was controlled with custom-written program running on a Raspberry Pi Pico, which communicated with the TDC-GPX2 via the SPI interface.

2. Introduction and theoretical background

Time-to-Digital converters are integrated circuits designed to measure very short intervals with extremely high precision-in the range of picoseconds. Microcontroller timers are often limited by their clock frequency, a TDC uses specialized delay lines and interpolation techniques to achieve finer resolution.

The TDC-GPX2 used in this project measures the time between a START event and one or more of STOP events. For each measurement the device outputs two values:

- REFNUM-the number of reference-clock periods counted between START and STOP
- STOP timestamp-the fine-resolution interpolated time within the last REFCLK period.

The TDC-GPX2 requires a stable REFCLK input to operate correctly. The reference clock defines: the coarse timing resolution, internal timing, stability and repeatability of the measurements. Without a valid REFCLK, the device cannot produce meaningful timestamps.

Because the TDC provides both START and STOP timestamps it can measure:

- Pulse width (difference between two STOPs),
- Pulse distance (difference between START and STOP),
- Multi-channel timing relationships.

3. Initial Measurements

The initial measurements were performed using the manufacturer's software provided by ScioSense. This software allows direct configuration of the TDC-GPX2 and visualization of the results.

The result of the TDC measurement can be presented using a histogram-with each part bin corresponding to a specific time interval and the height of the bin representing the amount of samples within that interval. Ideally, the histogram should be as close to the normal distribution as possible.

Several example histogram obtained during these measurements are shown below.

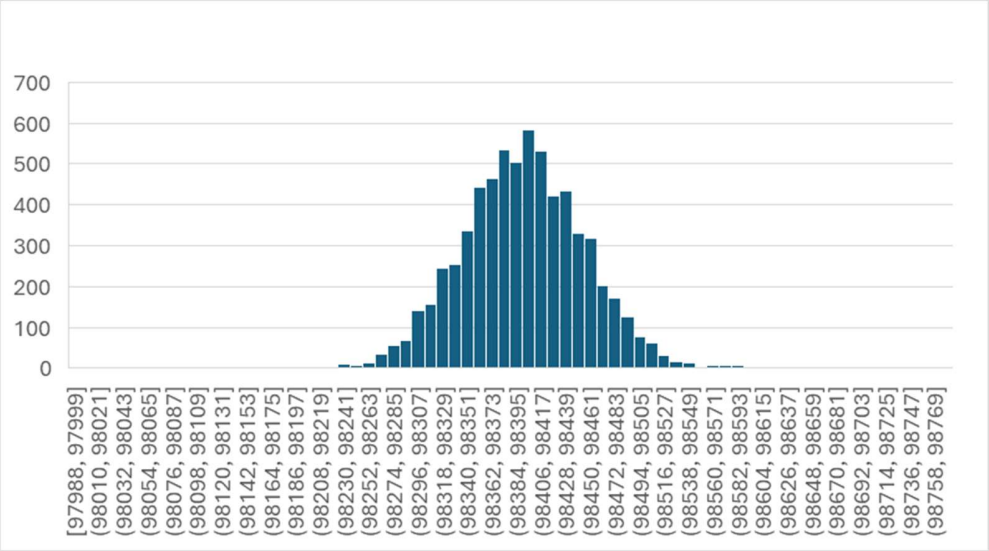

Figure 1: Histogram with duty cycle of the signals 50% and 80%

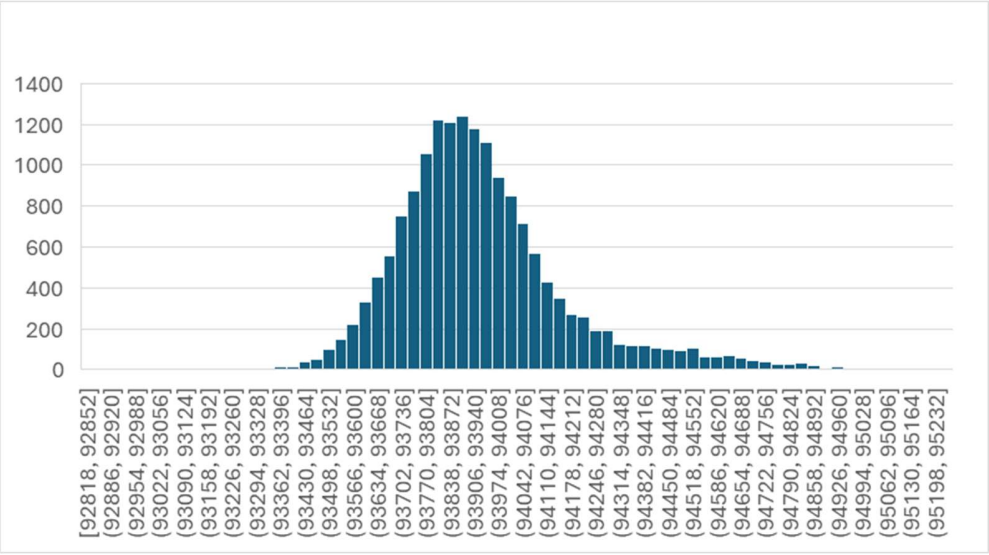

Figure 2: Histogram with duty cycle of the signals 50% and 90%

4. Custom control program

After verifying the operation, a program was written to operate the device independently. The program runs on a Raspberry Pi Pico and communicates with TDC via the SPI interface.
The main objective was to replicate the basic functionality of the manufacturer's software, while providing greater insight into the internal configuration and behavior of the TDC.

4.1 Configuration and communication

The Raspberry Pi Pico was configured as the SPI master, while the TDC operated as SPI slave. The program implements SPI routines responsible for:

- Writing configuration registers to TDC-GPX2,
- Reading back and verifying configuration data,
- Triggering measurement cycles,
- Reading measurement results from the device.

To ensure reliable operation, every configuration write is followed by a verification step. This allows for a detection of errors or conflicts before measurement begin.

4.2 Configuration Interface and Validation

A command-line interface (CLI) was implemented to allow interactive configuration of the TDC-GPX2. Through this interface, the user can enable or disable:

- STOP input channels,
- HIT_ENA signals,
- The reference clock (REFCLK),
- High resolution (HIRES) modes,
- FIFO operating modes,
- The internal crystal oscillator (XOSC),
- CMOS input mode.

Before applying the configuration, the program performs an automatic validation step. This validation checks for illegal or conflicting settings (mentioned in the manual), including:

- STOP channels enabled without corresponding HIT_ENA signals,
- Missing or invalid reference clock configuration,
- Incompatible combinations of HIRES and channel-combine modes,

Only configurations that pass all mandatory checks are written to the device.

4.3 Measurement Control and Data Acquisition

After successful configuration, the program initiates the measurement process and continuously monitors the interrupt output of the TDC-GPX2. Additionally, the program allows user to pause and resume measurement.

4.4 Results of the measurements

Analogically, the data can be presented using histograms. The results match the expected distribution shape, indicating stable and repeatable timing measurements.

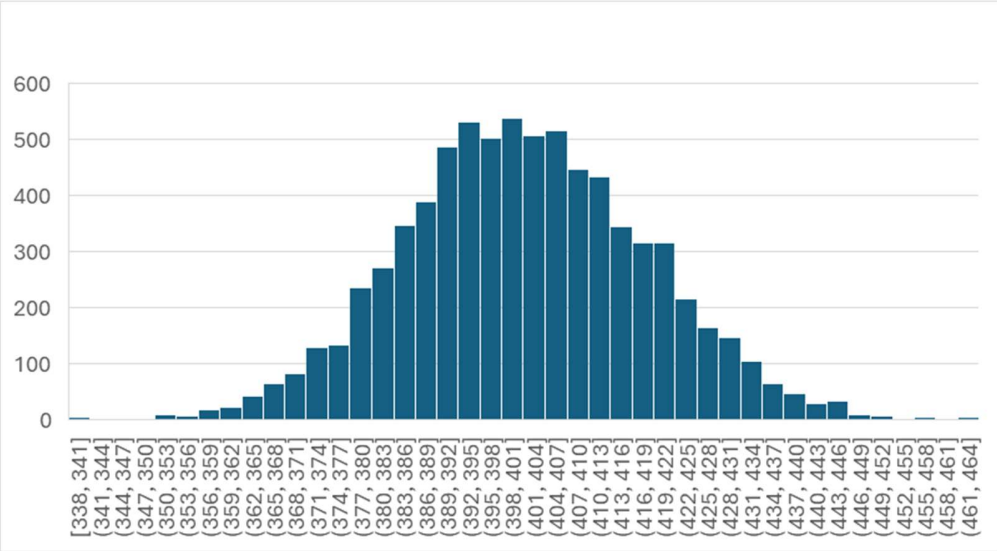

Figure 3: Histogram with duty cycle of the signals 50% and 80%

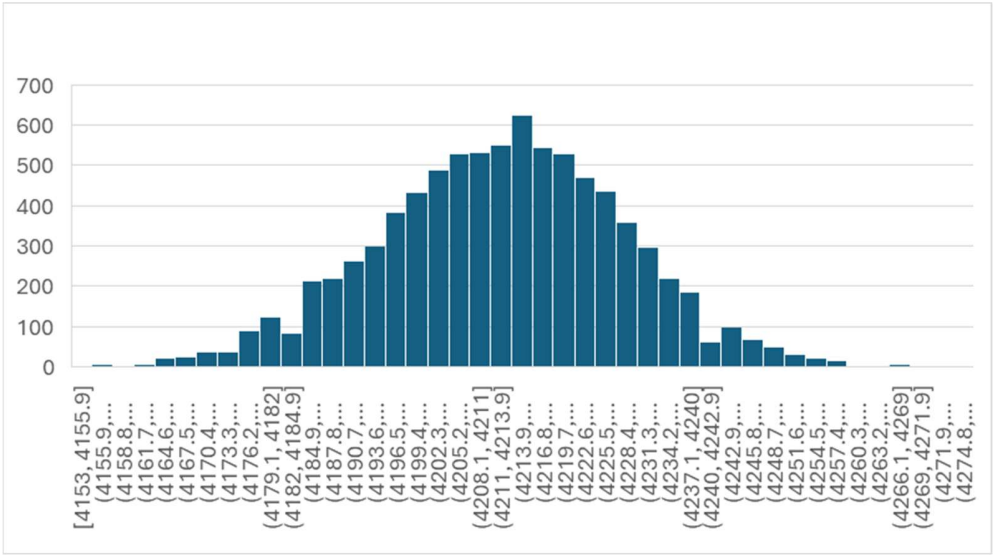

Figure 4: Histogram with duty cycle of the signals 50% and 90%

5. Python script implementation

One of the initial project objectives was to develop a Python script capable of communicating directly with the TDC-GPX2 evaluation board under linux, without relying on the Raspberry Pi Pico.
However, the TDC-GPX2 evaluation kit does not expose a simple USB interface, it uses a driver with a windows-only DLL provided by the manufacturer. The protocol used is not publicly available, so the board cannot be accessed as a standard USB device under linux. Reverse-engineering the protocol was considered, but it would be too complex for the limited time available for the project. For these reasons, the Python-based control interface was abandoned and only the Raspberry Pi Pico was used.

6. Conclusions

This project demonstrates the successful operation of the ScioSense TDC-GPX2 using both the manufacturer's software and a custom control program running on a Raspberry Pi Pico. The custom implementation allowed direct access to the configuration registers and provided greater insight into the internal operation of the device.
The results confirm that the TDC-GPX2 is capable of high-resolution time measurements when properly configured.