# Introduction

This is a high-level guide for experimenting with the MAXREFDES169. The kit can be used to learn about the MAX35104 time-to-digital convertor and its application in a typical gas flow meter design.

# Flowbody

The kit doesn’t come with a flowbody. You will need to manufacturer your own flowbody in order to work with the kit. This can be as simple as two ultrasonic transducers fitted to either end of a rolled-up paper tube. While such a simple configuration doesn’t allow for actual flow measurement, it will allow you to experiment and evaluate specific transducers in a zero-flow environment. This can be a very good way to get started if you are new to ultrasonic flow metrology.

# Firmware Functionality

The primary function of the kit firmware is to provide a user interface to the MAX35104 TDC which allows easier experimentation and learning. The interface is a serial command-line interface and requires a PC with a terminal emulation program, such as PuTTY, to be connected to the USB UART serial port provided by on-board PICO module.

This interface provides commands to configure the MAX35104 TDC registers as well as configure and execute data-collection activities. Time measurements made by the TDC can be reported to the serial interface and collected by the PC. This data can be used to develop and verify algorithms common to ultrasonic flow metrology such as transducer temperature compensation, flowbody compensation, and wave amplitude/phase tracking.

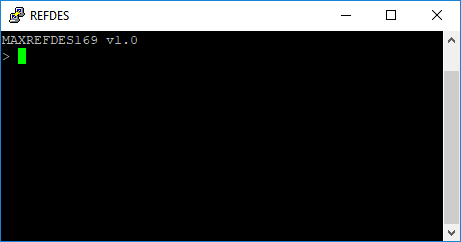


Figure 1-Command line interface via PuTTY

In addition to being a learning and evaluation platform, the firmware also serves as basis for production firmware specific to your design. The firmware includes an API for the MAX35104 that can be ported to another MCU platform. Wave tracking algorithms are also provided which can be used as-is or as a starting point for your own design.

Finally, the firmware could be used as a platform for end-of-line flowbody manufacturing calibration.

The firmware is an example of how to interface to the MAX35104, how to roughly structure data collection, compensation, tracking, and metering functions for those without a strong background in ultrasonic flow metrology.

# Getting started with the Command Line Interface

The kit is designed to be connected to a PC running a terminal emulation program, such as PuTTY. The USB connector on the kit provides the physical connection. The host PC must be properly configured to access the USB serial port. In Windows, the serial port appears as an mBed serial port in Device Manager.

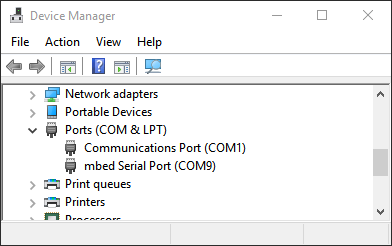


Figure 2- Configuring the USB serial port in Windows

The port should be configured to 115200 baud, 8 bits, no parity, and 1 stop bit. Be aware that the default for the mBed serial port is 9600. This will not work with the kit so the port must be intentionally configured for 115200.

The commands supported by the firmware can be grouped in to three different categories:

1. MAX35104 register access commands
2. Data collection/reporting commands
3. Platform commands

Direct register access is provided by commands that mirror the register bitfields defined in the MAX35104 datasheet. These commands allow you to set or query specific register bitfields in the TDC. For example, you can set the pulse launcher frequency via the dpl register with the ‘dpl=X’ command, where X is the desired frequency. The ‘dpl?’ command will return the current value of the register.

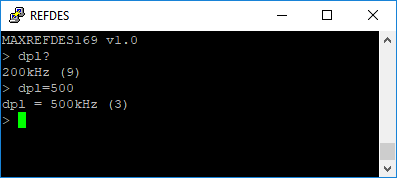


Figure 3-Using the 'dpl' register access commands

Data collection/reporting commands enable the period collection and reporting of time data from the TDC. This allows for data collection for analysis on a PC using a program like Matlab or Excel.

Platform commands are convenience functions like saving TDC register configuration to flash for later use.

The ‘help’ command lists all commands with a short description. For detailed information, review the source code in com.c. For more information about the register access commands, review the MAX35104 datasheet.

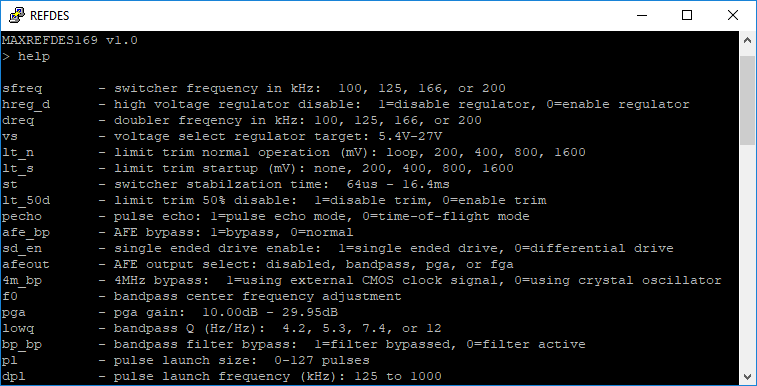


Figure 4-A portion of the result of the 'help' command

# Configuring the MAX35104

The process of configuration requires only that you know the resonate frequency of the transducers being used and the approximate time-of-flight of the ultrasonic wave from one transducer to the other. With this information in hand, an iterative process of trial-and-error using an oscilloscope attached to the kit is recommended.

1. Set the dpl register to correspond to the resonant frequency of the transducers. For example, with 400kHz transducers use ‘dpl=400’.
2. Calibrate the bandpass filter using the ‘bpcal’ command. This command should always be used after setting the dpl register.
3. Set the pulse launch count to a value in the 10 to 20 cycle range. ‘pl=15’, for example.
4. Set the measurement timeout it maximum value with the ‘timout=16384’ command
5. Set the comparator offsets to their maximum value. ‘c\_offsetup=127’ and ‘c\_offsetdn=127’
6. Set the PGA to a median value. ‘pga=20’
7. Set the excitation voltage to median value ‘vs=20’
8. Start periodic data collection with the ‘tofsr=10’ command. This performs 10 time-of-flight measurements per second
9. Optionally save this configuration with the ‘save’ command. Doing so will cause this configuration to be loaded at subsequent system resets.

At this point the firmware is initialing measurement commands at 10Hz and you can use an oscilloscope to observe the transducer signals, the output of the receiver AFE, and the output of the time-measurement comparator. These signals can be accessed via the test points on the right edge of the board.

The red LED, D3, is used by the firmware to indicate that a measurement has timed out. If illuminated this indicates that there is something wrong with the configuration, the transducers, or the connection to the transducers. Use the oscilloscope to determine the cause. Refer to the MAXREFDES169 and MAX35104 datasheets to understand the details of the TDC waveforms and their relation to configuration registers in the TDC.

From here the process is iterative. Observe the behavior of the testpoint signals and adjust the relevant TDC registers. Repeat until a stable and consistent measurement is made (no time-outs). Generally, register tuning is limited to the PGA setting and the comparator offset settings.

In addition to the periodic measurement test described above, a one-shot test can be performed using the ‘tof\_diff’ command. To use this command, first ensure that periodic measurements are disabled by using the ‘tofsr=0’ command. The output of the ‘tof\_diff’ command gives returns readable measurement data that might be useful to fine-tune comparator offsets and the PGA.

Configuring the MAX35104 for the first time can be difficult without a good understanding of ultrasonics and MAX35104 operation. It is highly recommend to carefully read the MAX35104 datasheet and related applications notes to establish foundational knowledge.

# Exporting Data using ‘report’

The command-line interface provides the ‘report’ command which can be used to log time measurement data for analysis by an external tool like Matlab or Excel. Such a mechanism is required due to the algorithmically intensive nature of ultrasonic flow metrology. Compensation of the transducers and the flowbody necessitate product-specific algorithms. These algorithms are typically developed based on experimental data collected across a product specific flow and temperature range.

Before using the ‘report’ command, use the ‘tofsr’ command to set the sampling frequency.

The ‘report’ command takes several options that provide different types of data. For low-level hit register data, use ‘report tof’. This can be useful to enhance the wave tracking algorithms provided by the firmware and other algorithms that use multiple hit registers as an input.

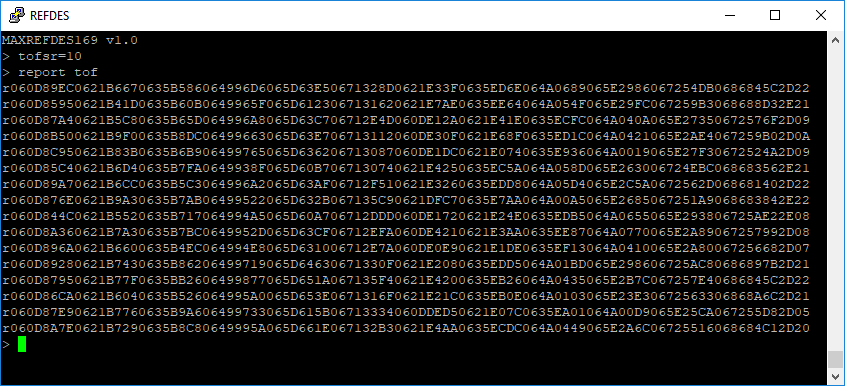


Figure 5-Output of 'tof report'

The format of the data is essentially a record identifier ‘r’ followed by a set of 32-bit hexadecimal values as follows:

1. Up hit register time-of-flight values 0 through 5, 32-bits (8 characters) each.
2. Down hit register time-of-flight values 0 through 5
3. Up t1/t2 ratio, 8-bits (2 characters).
4. Down t1/t2 ratio

For parsing details, refer to the Matlab script, read\_log.m, which accompanies the firmware.

The ‘report’ command also can return a more wave tracked form with the ‘tracked’ option. This option uses the wave tracking algorithm to maintain lock on a particular wave in the presence flow noise.

Temperature measures can be added to any report by using the ‘temp’ option with the ‘report’ command. Temperature measurements require an external 1K RTD sensor not included in the kit.

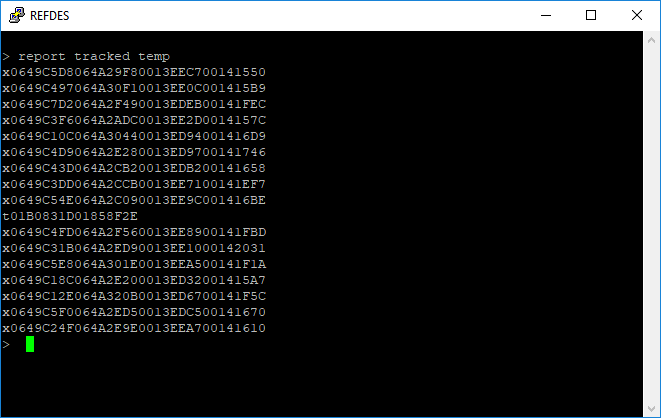


Figure 6- tracked time-of-flight with temperature

Temperature measurements require setting the TOF measurement to temperature measurement ratio with the ‘tempr’ command. The argument specifies how many TOF measurements to perform for each temperature measurement. For example, ‘tempr=20’ will perform 1 temperature measurement for every 20 TOF measurements. If ‘tofsr=100’, then the temperature sampling frequency would by 5Hz.

Temperature records are identified with a ‘t’. Two 32-bit hexadecimal values follow. The first is the discharge time for the external RTD/thermistor and the second is the discharge time for the on-board 1K ohm resistor. The ratio of these values can be used to calculate temperature. See the Matlab script, ratio2celcius.m for implementation details.

# Ultrasonic Flow Algorithm Development

Ultrasonic flow meters typically require at least four signal processing algorithms:

1. Waveform disambiguation
2. Waveform tracking
3. Transducer temperature compensation
4. Flow compensation

These algorithms are detailed in the following sections.

## Waveform Disambiguation

The kit firmware implements rudimentary waveform disambiguation and tracking. Transducer temperature compensation and flow compensation are not implemented as they are highly product specific. However, stubs have been included to allow for the easy addition of these algorithms. See flowbody\_transducer\_compensate() and flowbody\_flow\_sos() functions in flowbody.c

A typical product development task is to use the kit firmware to collect transducer and flowbody data across flow and temperature to produce compensation algorithms/coefficients to be implemented in flowbody.c

Waveform disambiguation addresses the problem of deciding which wave in the received signal is the correct wave to frame a time measurement. Being off by even one wave will cause huge measurement errors. When you consider the propagation time of the ultrasonic signal across temperature and flow rate, you will find that identifying the proper wave to be very challenging and require system-specific knowledge to accomplish.

A typical solution generally requires starting form a known state--zero flow at a certain temperature, for example and tracking incremental changes from there. However, this simple algorithm may not work for all products. Some products may require a ‘setup’ mode to specify starting conditions, other products are constrained enough to be able to rely only on signal amplitude. Some products may require measurements from other sensors to assist in disambiguation, a temperature measurement for example. Finally, some products might require multiple transducer pairs with multiple MAX35104 TDC’s to disambiguate across very large propagation ranges. Put simply, waveform disambiguation is highly product specific.

## Waveform Tracking

Waveform tracking follows from disambiguation. Once the measurement wave is identified a tracking algorithm is used to make sure all measurements are based on the same wave of the ultrasonic signal. The general idea is that adjacent measurements, with a high enough sampling frequency, will have a similar time measurement. Large shifts in time generally mean that the measurement wave in either the up direction or the down direction (or both) has been misidentified and the sample must either be corrected or discarded.

There are two facets to waveform tracking: 1) Amplitude adjustment (software AGC), and 2) Phase adjustment. Amplitude adjustment is an automatic gain control algorithm that uses the t1/t2 ratio as a proxy for waveform amplitude. This algorithm adjusts the comparators offsets according to relatively slowly changing signal levels.

Phase adjustment works cooperatively with amplitude adjustment. In cases where the amplitude changes too fast to be tracked, a phase error detecting and adjustment algorithm relocates the measurement wave within a certain range based on the assumption that the time-of-flight from one measurement to another (given an adequate sampling frequency) is less than one transducer oscillation period.

See wave\_tracking.c for an implementation example of amplitude and phase tracking.

The command line interface allows control of wave tracking via the ‘rtrack’ command. This command assigns a value that corresponds with the optimal point at which to set the comparator thresholds relative to the received wave’s peak. Normalized between 0 and 1, values of 0.6 to 0.8 typically yield good tracking results. ‘rtrack=0’ will disable tracking which is useful when experimenting with the comparator offsets manually with the ‘c\_offsetup’ and ‘c\_offsetdn’ commands.

## Transducer Temperature Compensation

Transducer temperature compensation adjusts time-of-flight values to remove directionally specific delays caused in by imperfections in the transducers. Specifically, oscillation frequency drift due to absolute temperature and the temperature difference of the internal piezo material and ambient.

Transducer temperature compensation can be as simple as a straight-line compensation using the aggregate oscillation period of the transducers (the difference from one hit value to the next defines the period of oscillation. See AN6631 for detailed information about transducer compensation.

## Flow Compensation

Flow compensation is necessary as the velocity of gas through the flowbody relative to ultrasonic propagation time is not linear due to imperfections in the mechanical design of the flowbody. Compensation generally takes the form of table of points that can be provide interpolated coefficients to transform raw TDC measurement times into linear flow velocity with defined units.

The process of generating this table typically involves moving gas through the flowbody under test at known rates using a precision gas flow regulator. Unitless “raw” flow is calculated directly from the TDC’s measurements are recorded for each flow rate. These values are then used as the mathematical basis for the coefficients of the table.

This method was used to obtain the default flow compensation table for the kit firmware and a prototype flowbody. A Parker XXXX Digital Flow Regulator under the control of Matlab scripts was used to generate These scripts are included in the kit int the ‘math’ subdirectory. These scripts are offered as a starting point for you to generate a custom compensation method specific to your product.