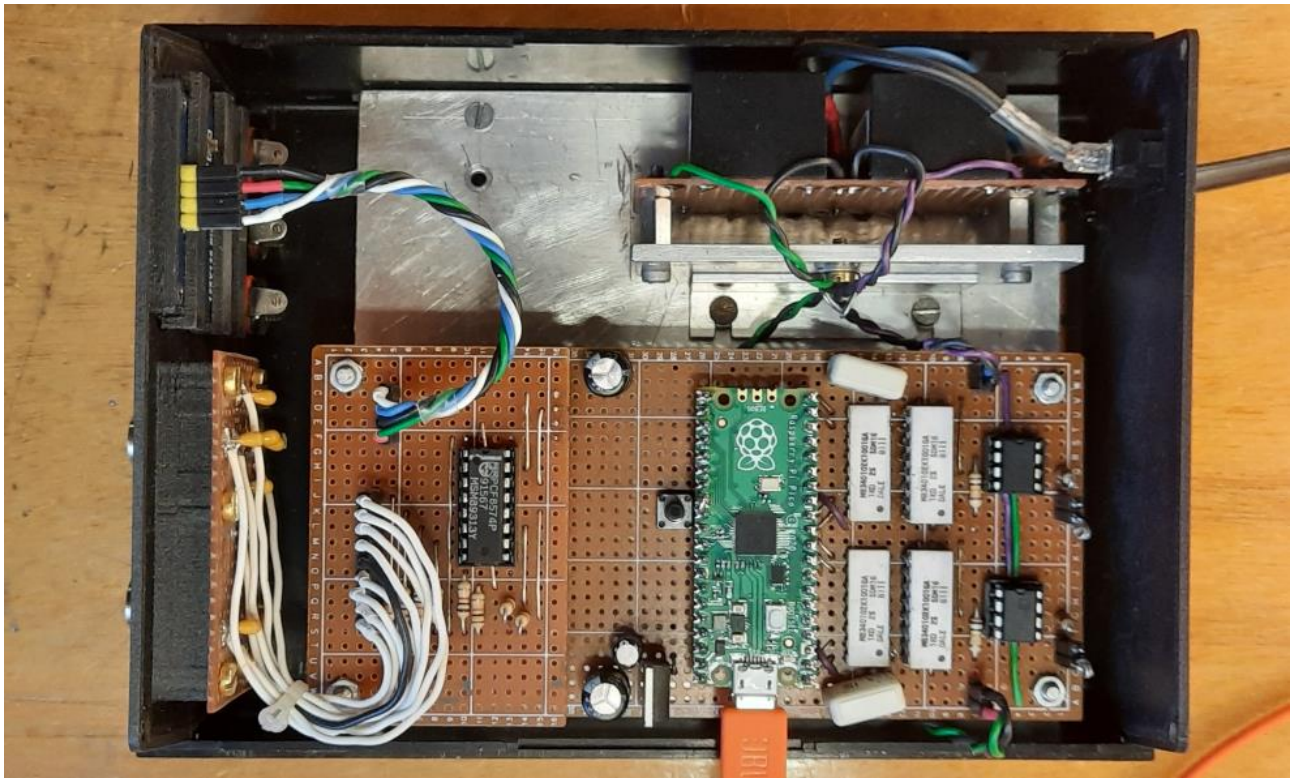
Micro WFG on a Pi-Pico

Arjan te Marvelde, Version 0.1 – February 2022 (initial version Dec 2021)



In this project, the fairly new Raspberry Pico (**RP2040**) is used to build a Signal Generator. After having gained some experience with my uSDR project, where the multi-core and signal processing were the main subject, this µWFG combines the very nifty Programmable I/O (PIO) feature with the flexible DMA service.

The goal is to create the ability to play arbitrary waveform files to GPIO pins, in principle at a rate up to system clock speed, i.e. 125MHz. The 8-bit samples then are converted into an analogue signal by means of a resistor ladder network, which is buffered by two AD811 high-speed video Op-Amps.

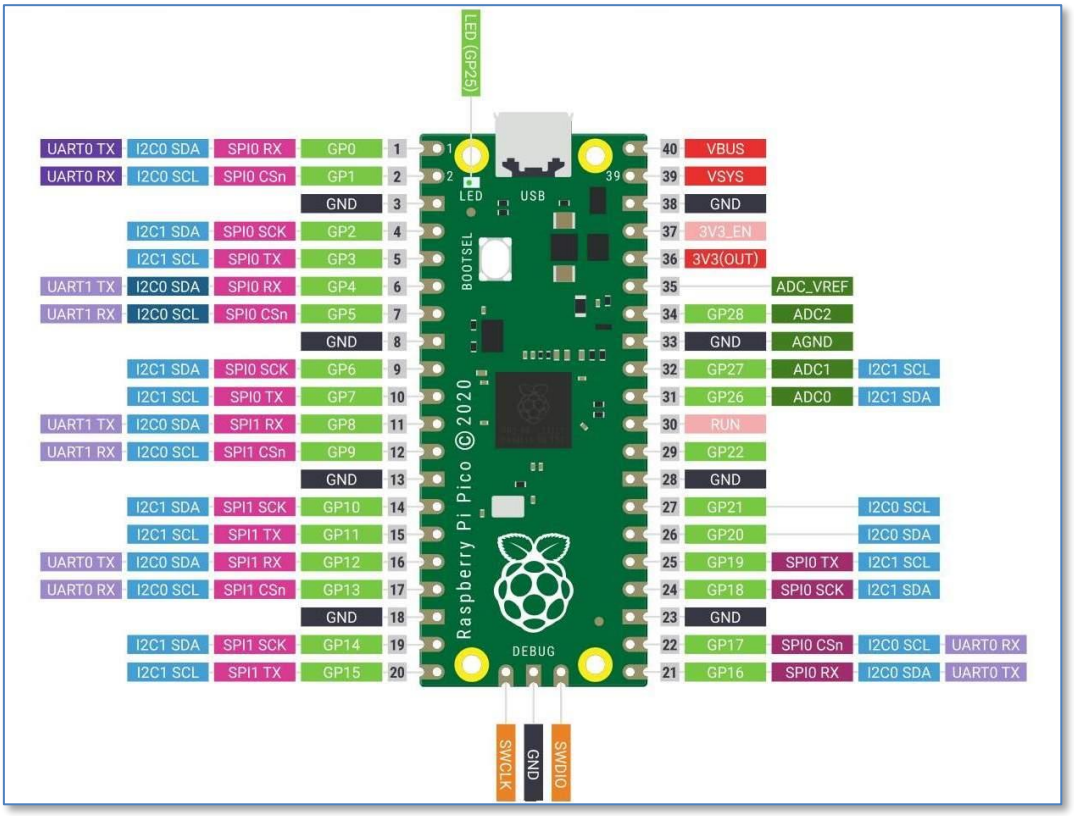
Since this works so well, the idea is to include some more test devices into this system. An obvious choice would be a component tester, for R, L, C as well as semiconductors. Examples are:

- Ardutester (https://github.com/kr4fty/Ardutester) or
- Transistortester (https://github.com/Mikrocontroller-net/transistortester).

Note to builders: This project is highly experimental and will remain a work in progress, I try to spend some time on it every now and then but it is just hobby. So, feel free to copy this project and add or change whatever you like. It is intended for experimentation and hence should not be considered as a flawlessly working kit.
This project is maintained on GitHUB: https://github.com/ArjanteMarvelde/uWFG-Pico

1 Raspberry Pi Pico (RP2040)

The Pi-Pico module is the heart of the implementation, so here is the pinout and the way it is used in this project.

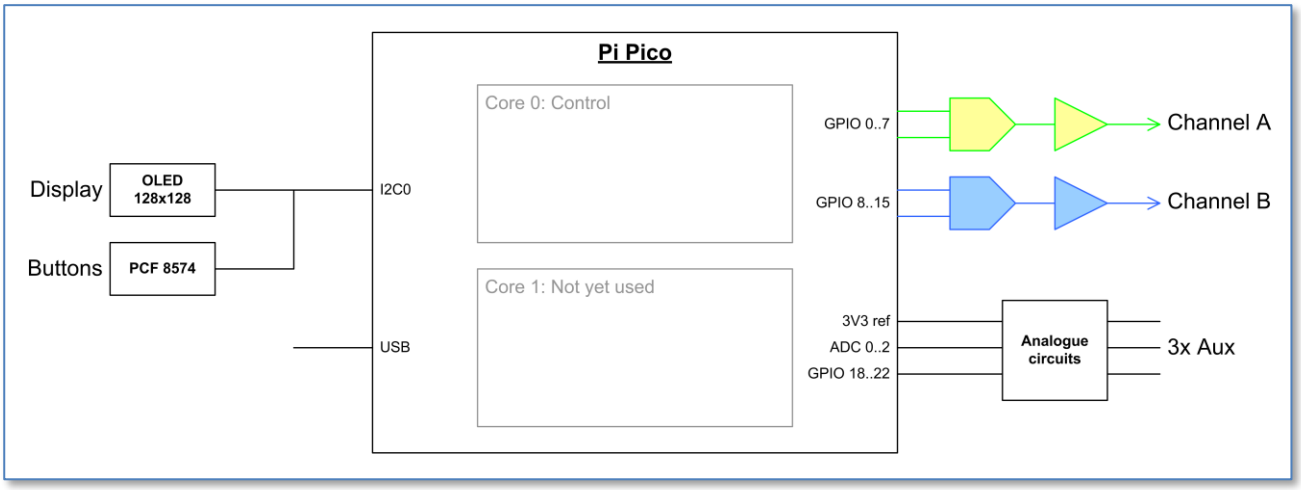


Pi Pico pin usage for uWFG project

Channel A – bit 0	GP0	Vbus		NC
Channel A – bit 1	GP1	Vsys		5V power input
	GND	GND		
Channel A – bit 2	GP2	3V3 en		NC
Channel A – bit 3	GP3	3V3 out		
Channel A – bit 4	GP4	Va ref		
Channel A – bit 5	GP5	GP28		ADC2
	GND	GND		
Channel A – bit 6	GP6	GP27		ADC1
Channel A – bit 7	GP7	GP26		ADC0
Channel B – bit 0	GP8	RUN		Pushbutton to ground for reset
Channel B – bit 1	GP9	GP22		Test stimulus
	GND	GND		
Channel B – bit 2	GP10	GP21		Test stimulus
Channel B – bit 3	GP11	GP20		Test stimulus
Channel B – bit 4	GP12	GP19		Test stimulus
Channel B – bit 5	GP13	GP18		Test stimulus
	GND	GND		
Channel B – bit 6	GP14	GP17		I2C0 – SCL
Channel B – bit 7	GP15	GP16		I2C0 – SDA

2 Principle of operation

The block diagram shows the processor board and several peripherals.



The architecture is quite simple, just the Pico and DA conversion. A serial interface is added via USB to enable file upload and control from a PC. The functionality is divided in three, a generator, a tester and a control part.

The **control part** merely feeds the generator and the component tester with the proper parameters, based on commands entered through the I2C connected user interface and display.

The **generator part** is somewhat more complicated, and consists of a PIO program that reads bytes from a FIFO and pushes the bits to the assigned GPIO pins. The trick is to keep the TX FIFO filled so no gaps occur. Luckily the output is only 8 bits wide, so the 4 in 1 speed ratio (32-bit FIFO) can be used to assure this. The FIFO is filled by means of a continuous DMA stream being supplied from the active waveform buffer. Continuity is accomplished by chaining two DMA channels, which trigger each other intermittently.

The **component tester** is an add-on function, which interfaces literally on the other side of the Pico board. It utilizes the three AD converters on GP26..28, as well as the remaining GPIO's GP18..22, to execute several test scenarios.

Bulk data exchange, such as loading auxiliary wavefronts but also the monitor interface use the USB connector on the Pico board.

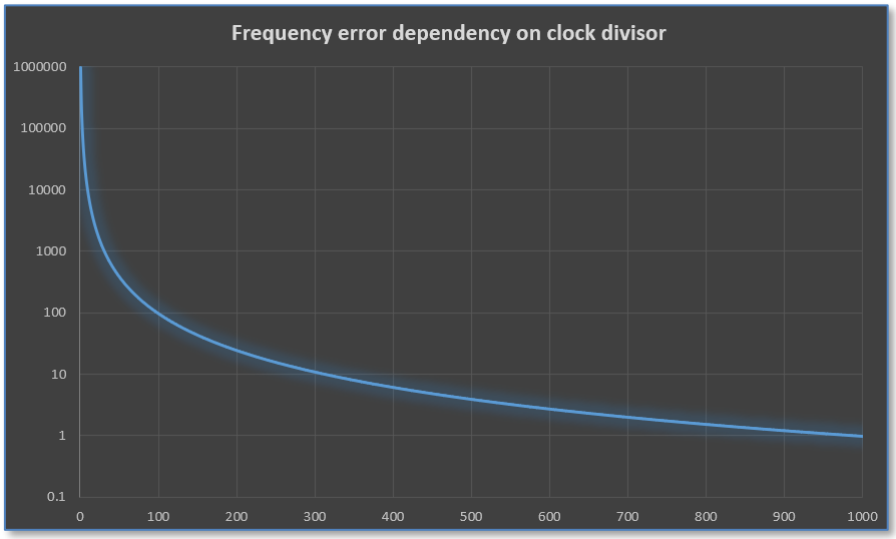
2.1 Waveform Generator

The waveform generator merely outputs a samples file on GPIOs, two channels of 8bits wide. The parameters that can be varied by a user are limited to waveform shape definition and playback frequency. Regarding the waveform, a selection can be made from predefined shapes, a pulse shape can be adjusted in terms of duty cycle and rise / fall times. Also, a random user defined file representing any shape can be uploaded. The output sample rate can be adjusted up to a maximum of the system clock (overclocked: ***sys_clk***=250MHz) and down to a minimum of ***sys_clk***/65536, so approximately 4kHz. With a maximum sample buffer size of 2000 a lowest frequency of 2Hz can be achieved this way.

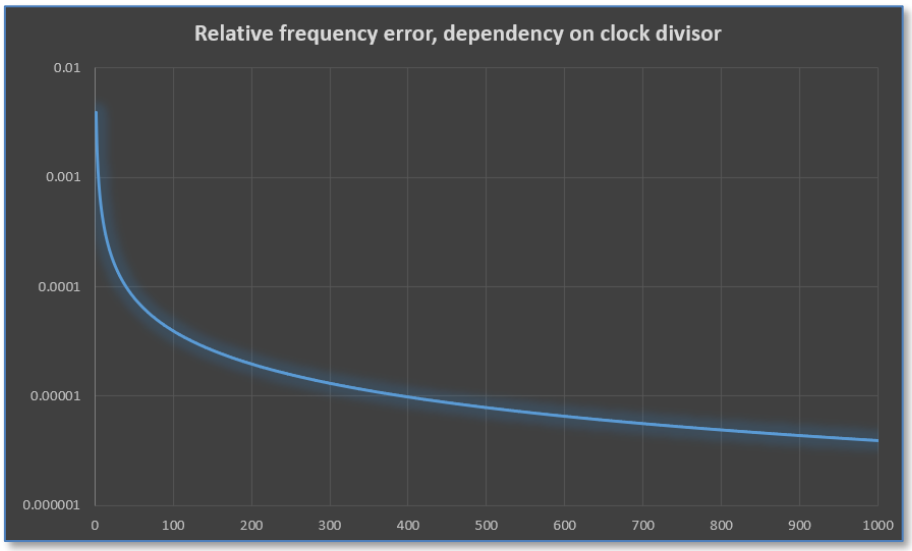
2.1.1 Precision

The *precision* of the clock division factor is 1/256, starting at 1. This implies that the higher sample rates (i.e. low division factors) will be have relatively coarse adjustment and may also suffer more from jitter. To meet the timing of the generator mechanism, a minimum sample file size is required of about 16 samples (4 x 32bit word, which is the bare minimum). When high frequencies are needed, it is therefore better to duplicate the waveform, although there is also a limit on waveform definition. The upper limit would be a square wave with a frequency of half ***sys_clk***, provided that the analog circuitry can handle this.

So, the problem at hand is to find the best combination of sample rate and waveform definition depending on user input. This input will be waveform selection and duration setting for simple waveforms like sine, square, sawtooth and triangle. For pulse types the rise and fall times as well as pulse duty cycle can also be set.



The strategy will be to maximize the number of samples in a waveform definition, where still a sys_clk divisor can be selected that provides reasonable precision. The maximum sample clock error is determined by the divisor step size, but it drops quite rapidly. With a ***sys_clk*** of 250MHz, the error drops below 100Hz when the divisor is larger than 100 and when the divisor is 300 the error is still only about 10Hz. But for small divisor values the step size can be up to 1MHz.



The relative error, as related to the resulting sample clock, sinks below 0.1% already at division factor 4, at factor 40 the relative error drops below 0.01% and at factor 400 even below 0.001%. The relative error of course

So, for a reasonable frequency error the divisor should be higher than 100, yielding a sample clock of 2.5MHz. Assuming 24 samples, the minimum waveform duration is 10usec with a maximum error of 1nsec.

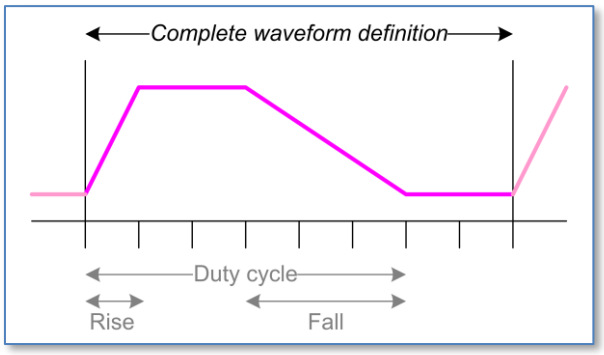
Accepting a bit higher frequency error, a divisor limit of 10 could be chosen, where the minimum waveform duration is 1usec and the maximum error also about 1nsec.

2.1.2 Waveform set up

The user can select from various types of waveform:

- Square wave
- Triangle wave
- Sawtooth
- Sine wave
- Pulse

The selected waveforms need to be further specified with a duration time. Amplitude is always maximum (appr $3V_{pp}$) but can be adapted in the analogue buffering circuits.



The pulse waveform is user definable, and apart from the total duration it is specified by the duty cycle, and rise/fall times. These phases are given relative to the full waveform duration, for example an upgoing sawtooth would be defined by 100% duty cycle, 100% rise time and 0% fall time.

After selection of the waveform parameters, the sample array is generated and handed to the generator to play on channel A or B.

When aiming at certain division factors (***sys_clk*** = 250MHz), the following overview gives the approximate relation with wave duration and required number of samples in the buffer:

T_{dur} [sec]	Div = 1	Div = 10	Div = 100	Div = 1000	Div = 10000	Div = 50000
10^{-7}	25					
10^{-6}	250	25				
10^{-5}	2500	250	25			
10^{-4}		2500	250	25		
10^{-3}			2500	250	25	
10^{-2}				2500	250	50
10^{-1}					2500	500
1						5000

So, the choice of buffer length in general can be chosen fairly large, unless the duration drops below 10usec or so. In general, clock divisors lower than 10 should be avoided unless timing accuracy is less an issue. The longer durations require large buffers anyway.

2.2 Component tester

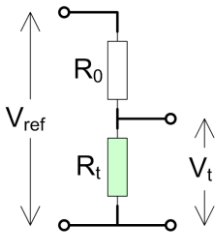
To test components up to three leads, three test points are needed. The three ADC channels will be hooked to these test points, because that is where voltages need to be measured. The stimuli are controlled through the remaining GPIOs.

Before doing the actual measurement, the tester needs to decide what kind of device is plugged in. This process uses a certain ordering of stimuli to go through a sorting tree. When it is clear what the device is and how it is attached, the actual measurement can start.

2.2.1 Two lead devices

2.2.1.1 Resistor

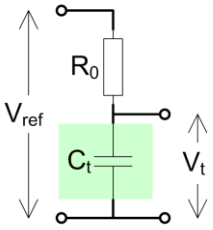
Testing Resistors is quite straightforward, just apply a known voltage V_{ref} over a divider with a known resistor R_0 and the resistor to be tested R_t. The test voltage V_t over R_t is measured and the value of R_t can easily be deduced.

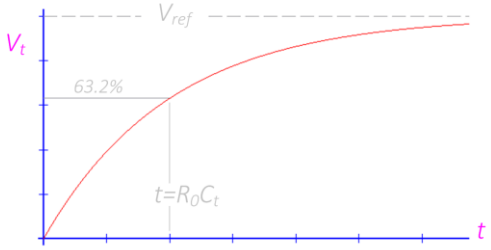


Some attention needs to be given to the desired accuracy; the ADC has 12 bits, so precision is V_{ref} / 4096. A too large value of R_0 will lead to low V_t and hence coarse resolution, and with a too low value V_t will hardly change anymore. So ideally the values of R_t and R_0 should be close, i.e. V_t should be targeted around half the reference voltage.

2.2.1.2 Capacitor

A capacitor can be tested in combination with a reference resistor as well, forming an RC network. The rising voltage as function of time can be measured and from this the capacitance deduced. There are however limits as to how accurate this process is.





The ADC can do a measurement every 2µs, after which the result is compared with a predefined level, more precisely the level that is reached after the RC time is over, 63%. If V_{ref} represents the full ADC range, the inaccuracy is inherently determined by the 2µsec conversion time and the $V_{ref}/4096$ sample precision. This is best case…

So, to obtain an acceptable relative accuracy we cannot change the sample precision, but we can stretch the RC time by selecting a proper R_0 value. For a 1% accuracy the RC time, being the total integration time, needs to be at least 200µsec. With a resistor of 1MΩ this would yield a lower capacitance boundary of 200pF.

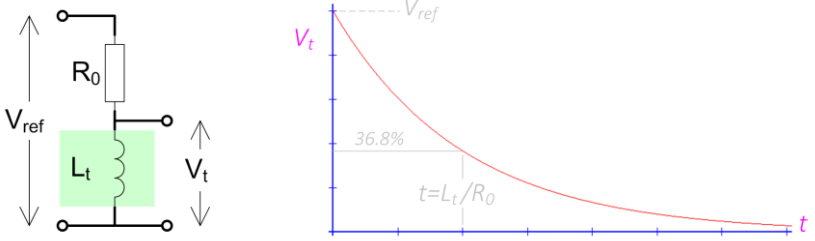
Integration time with the same resistor would be 1sec with a 1µF capacitor, that is about the time we would switch to another integration resistor of e.g. 1kΩ.

Another interesting parameter to measure is the parasitic resistance (Equivalent Series Resistance, ESR). The measured voltage V_t will step down a little after the charging current is stopped. This step represents the voltage drop over the ESR while charging, so it is directly related to R_0. When indeed charging is done until the RC time, the charging current was $(V_{ref}-V_t)/R_0$ and the ESR can be derived from this and the step in V_t.

Another way of measuring this is with a high frequency square wave.

2.2.1.3 *Inductor*

An inductor could in principle be measured in the same way a capacitor is done, but the V_t -graph is reversed. Instead of voltage build-up there will be current build-up, so it starts with full V_{ref} which drops to 36.8% after L/R time.



In contrast with the capacitance measurement, here the R_0 value should be kept (very) low for higher accuracy. With a 1kΩ value and 200µsec integration time the lower limit for L is in the order of 200mH. The value of R_0 cannot be lowered much more, because of the output current limitations of the Pico (max 12mA).

So for lower inductance values (and really also capacitance), a different method should be utilized.

2.2.1.4 *More accurate C and L methods*

Sine wave response

Since we have a waveform generator, we use it to measure more accurately small C and L. Instead of using a constant V_{ref}, this will be replaced with the output of the WFG; V_{gen}. The frequency of V_{gen} can be adjusted up to about 1MHz while maintaining a fairly good sinewave (at 125 MHz sample rate).

The relation between V_t and V_{ref} is given by:

$$\frac{V_t}{V_{ref}} = \frac{X}{\sqrt{{R_0}^2+X^2}}$$

where voltages are peak values and X is the reactance of either C or L:

$$X_L = 2 \cdot \pi \cdot f \cdot L \qquad \text{and} \qquad X_C = (2 \cdot \pi \cdot f \cdot C)^{-1}$$

Let's say we go for 50% voltage ratio, then we can solve the relation for X and R_0:

$$X = \frac{1}{\sqrt{3}} \cdot R_0$$

For a few well selected values of R_0 we can tune the frequency until the 50% voltage ratio is reached. Knowing frequency and R_0 the capacitance or inductance can be deduced. Some approximate values:

Freq	R0	C	L
1MHz	1MΩ	0.3pF	90mH
1MHz	1kΩ	300pF	90µH
1kHz	1MΩ	300pF	90H
1kHz	1kΩ	300nF	90mH

As can be seen, the method cannot be scaled down to really small inductances. Changing the voltage ratio to 10% only improves this a factor 5 or so, increasing resistance over 1MΩ also is not realistic.

Triangle wave response

Alternatively, using the basic inductance and capacitance equations:

$$V_L = -L \cdot \frac{\partial I}{\partial t} \qquad \text{and} \qquad I_C = C \cdot \frac{\partial V}{\partial t}$$

we could derive the values when a constantly changing current or voltage can be created. This could be implemented by using a triangle waveform instead of a sinewave for V_{ref}. In the equation the current I is given by:

$$I = \frac{(V_{ref}-V_t)}{R_0}$$

So, in order to get V_t=1V for a 1µH inductor, the current should change with a rate of 1MA/s, or scaling it to higher frequency, 1A/µs. This seems quite challenging too, but considering a lower value of V_t and a bit higher frequency this may just be on the edge. Still, 100nH coils are out of league.

Boxcar detection

One parameter that a microprocessor can determine fairly accurately is *delay*, effectively increasing the resolving power of AD conversions for *repeating signals* of known frequency. For low values of C, and especially L, the 2µs ADC resolution of the Pico is not good enough, but by repeating the signal and increasingly delay the start of a conversion, a more accurate picture can be built. In principle an accuracy of 1/48 usec, which is an improvement of about 2 decades.

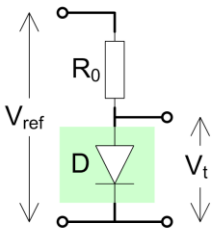
See for example http://pa3fwm.nl/technotes/tn11b.html for more background.

So, if L/R or RC time is detected to be smaller than 200µs, this technique can be applied instead. We need to start a pulse and immediately also a delay timer. When the latter runs out, the sample is taken and a conversion is done.

The pulse length must be so large that the V_t value is almost zero, so 1ms (more than 5·RC or 5·R/L) should be fine for sub 1% error. Then the same time must be spent to account for the counter-pulse and a new one can be started. The ADC delay for a repeating free-running conversion can be set to a multiple of 48MHz clock, or 20.83333… ns. The clock divisor must be set at a value larger than 96000 to get more that 2ms, so 96001 would in principle yield a 21ns resolution. For a 1% time resolution the minimum RC or L/R time now becomes 2µs, i.e. for 2mH it is still too much…

2.2.1.5 Diodes

Diodes share the feature that they conduct in one direction and behave differently in the other.



The voltage drop in forward direction (V_f) is determined by the manufacturing technology. Schottky diodes have a V_f of lower than 0.3V, Germanium diodes just over 0.3 and Silicon diodes closer to 0.7V.

Measuring the diode in reverse direction, usually the measured voltage will be almost equal to V_{ref}, , a drop will represent the (tiny) reverse leakage current. For Zener diodes with a breakdown voltage lower than V_{ref}, the measured voltage will be around this value. Measuring a larger range of Zeners therefore requires a higher supply voltage and also some means to divide the breakdown voltage to measurable values (lower than V_{ref}).

All diodes have a capacitance which is reverse voltage dependent, but Varactors (or Varicaps) have a large dependency. Usually this also requires a larger voltage range to measure

2.2.2 Three lead devices

2.2.2.1 Transistors

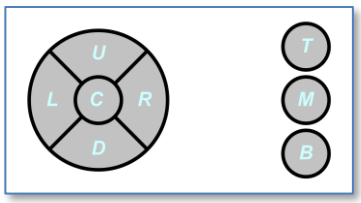
Binary Junction Transistors (BJTs) basically come in two sorts, NPN and PNP.

2.2.2.2 JFET

2.2.2.3 MOSFET

2.3 User dialogues

The user interface provides two BNC connectors for channel A and B, and three auxiliary banana sockets for other measurements. The operation has a set of buttons and a display.



The HMI provides dialogues for setting the signal channel A and B output parameters:

- Waveform selection
- Waveform shape control
- Duration of a cycle

Also, it provides dialogues for carrying out the measurements on devices connected to the auxiliary interfaces.

- Type of measurement (L, C, R, or semiconductor)
- Initiate the measurement

The idea is to use the three buttons on the right (T, M, B) to select the dialogue, and to use the navigation pad to change the various settings.

3 Software

Overview of the files:

uWFG.c	System initialization and main loop that calls various handlers.
gen.c	Generator interface, to set-up the PIO data feed.
wfgout.pio	PIO instructions to output data on GPIO channels A and B.
lcd.c	Interface to OLED display, character based or graphic.
hmi.c	User interface, input buttons and dialogues.
monitor.c	A command shell running on the stdio UART.

3.1 Main (uWFG.c)

The main() function merely starts up several system resources and initializes some of the SW components. Then it ends in a loop triggered by a loop timer set at about 50msec. This loop calls the monitor and hmi evaluation routines, in fact polling for input.

This file also defines a timer callback function that toggles the LED GPIO.

3.2 Generator (gen.c)

The generator component contains everything necessary to playback the waveform on the outputs. It provides the generator API function gen_play(), which copies the waveform parameters and samples from the given structure and sets up the DMA and PIO clock divider to obtain the desired waveform.

3.3 Component tester (rcl.c)

The tester component provides the routines that execute the tests that can be invoked through the HMI.

3.4 OLED display (lcd.c)

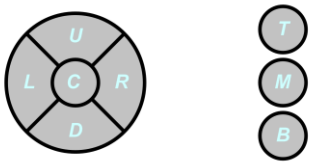
This file provides the interfaces towards the graphic 128x128 pixel OLED display. The basic action is outputting characters from one of the available fonts on a specific display location. Since the display is graphic, the location points at the left upper corner of the character field, where the size is determined by the font. Outputting a string basically does the same, but repeatedly while shifting the character field location in x-direction. Characters that fall outside the display area are not printed.

The same mechanism can be applied for graphical output, where a bitmap that is defined as a single character font can be written to the display in basically the same manner. Finally the interface provides some basic line drawing and window clearing functions.

All graphical interface functions can be in regular or inverse video, the two-pixel (greyscale) bytes are simply XORed with 0xff.

3.5 User interface (hmi.c)

The user interface loop consists of an event handler that is invoked from the main program loop. The first thing that is done is evaluate the button status through the I^2C register.



There are three auxiliary buttons (B, M, T) and four navigation buttons plus a center button (C, U, R, D, L), adding up to a full byte. Feedback to the user is given through the OLED display, which is also coupled via the I^2C bus (see lcd.c).

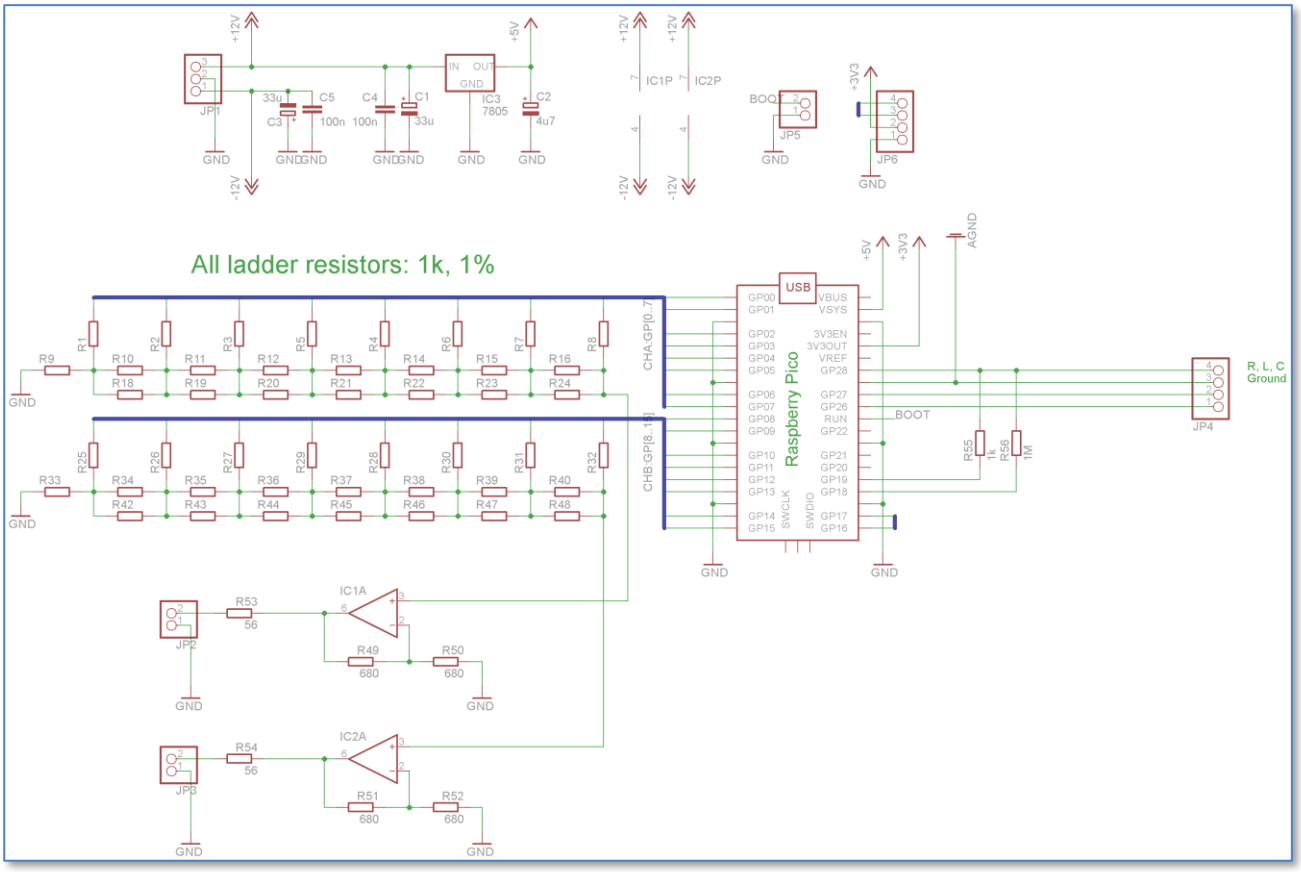
The top and middle auxiliary buttons select the channel A and channel B menus, while the bottom auxiliary button selects the menu for other measurements.

3.6 Monitor (monitor.c)

The monitor is a command shell that runs on a serial port TTY, connected to a PC over the USB interface.

The shell can be easily expanded. It consists of a routine that collects a string from the serial interface, splits it up into white-space separated substrings which can be processed by a commend. The first substring is the command, which is compared with strings in a lookup table. When a match is found, the corresponding routine is called with the argv and nargs as input. The working of the routine itself is up to the user.

4 Implementation



Schematic of the prototype. I have used high-speed video line drivers AD811 as analogue signal buffers. These OpAmps go up to 140MHz, and give a very good representation of the waveforms for the frequencies used.

5 Testing

A first breadboard prototype was built to test the software and to test the digital to analogue conversion. More specifically, it is important to investigate what the best match will be between GPIO output characteristics, ladder network impedance and buffer amplifier. For this reason, R-2R ladders are made with R=5kΩ for channel A and R=1kΩ for channel B. Without any special care for the buffer amplifier the resulting 1MHz square waves are shown below:

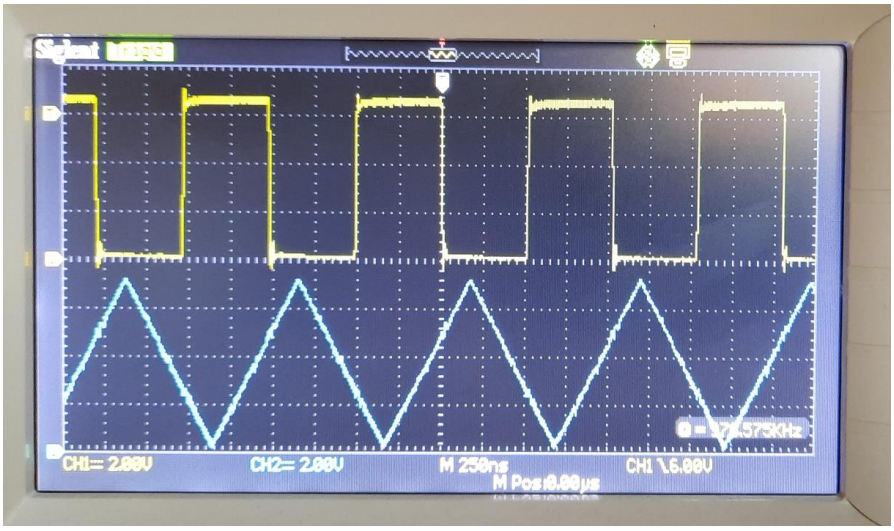
The top signal is the 5kΩ ladder, and the bottom the 1kΩ ladder. Clearly, the higher resistance network is more susceptible to parasitic capacitance, while the low resistance network can still be sufficiently driven by the Pico outputs.

When the frequency is increased, the longer rise and fall times in channel A result in reduction of the signal amplitude. Possibly, this can be compensated by increasing the buffer circuit input impedance, and make this frequency dependent.

Reducing the ladder resistor in channel A to 0.5kΩ makes the edges even steeper, while the Pico can still cater for the output current.

The second prototype has richer functionality and is built into a decent enclosure:

The HMI for channel settings is almost complete here for the generator part. In these images the channels A and B are set for 1usec period square and triangle waves.



The output amplifiers have been modified slightly with respect to the first prototype, in order to reduce overshoot and make waveform reproduction more accurate. The irregularities of the resistor network are nicely visible in the triangle wave, as is the presence of the 250MHz clock noise in the high level of the square wave. Smoothing out by low pass filtering doesn't seem to help this very much though. Also, note the output swing of about $7V_{pp}$, which is twice the Pico 3V3 supply (and output) voltage. Of course, this is due to the 2x amplification in the video amplifier.