Low Distortion DDS Signal Generator

This two-channel audio signal generator produces very low distortion sinewaves as well as triangle waves, square waves, pulse trains and noise. It has adjustable output frequency, amplitude and phase, plus sweep and pulse modes. Control is via a rotary encoder knob and graphical LCD screen with an intuitive graphical user interface (GUI). It's ideal for testing amplifiers, loudspeakers and all sorts of audio equipment, as well as for general purpose use.

by Phil Prosser

On my work bench, I find the most useful test equipment – after my multimeter – are a signal generator and an oscilloscope.

When testing analog circuitry, I find it very important to be able to 'stimulate' a circuit and then look to see how it responds.

In the past, I used simple oscillators for this job, usually Wien Bridge types due to their high performance at low cost.

But when testing speakers, it is very useful (in fact, almost essential) to be able to sweep the generated tone frequency. This allows multiple drivers and crossovers to be tested.

And when testing amplifiers and speakers, it is very handy to be able to generate short bursts of a tone with a silent gap. A simple oscillator can't easily do either of those things.

Burst tones serve a couple of purposes. Firstly, when testing a power amplifier driving a low impedance, it can be quite stressful on heatsinks, dummy loads and power supplies using a continuous waveform.

If you can generate, say, two cycles at 1kHz followed by a second of silence, you can see what the amplifier does at and near clipping and at high currents without really stressing things.

Also, when building loudspeakers, it is really useful to be able to generate tone bursts at and around the crossover point. This lets you set up a microphone to measure the time delay for the tone burst through a bass driver and a tweeter.

This is essential if you want to 'time-align' drivers in a speaker cabinet.

For some time, I used the free Audacity software on my PC to do these jobs.

But that set-up was a bit clunky, so I set about designing a more convenient hardware device that could do all this for me.

The device that I came up with, presented in this article, uses the same hardware as my DSP Active Crossover/Parametric Equaliser project that was published in the May, June and July 2019 issues (see: siliconchip.com.au/Series/335).

If you've already built that, it's simply a matter of reprogramming the microcontroller to perform these waveform direct digital synthesis (DDS) functions.

If building it from scratch, you can build it as described in those earlier articles, although you only need one of the stereo digital-to-analog converter (DAC) boards (not two) and you don't need the analog-to-digital converter (ADC) board at all.

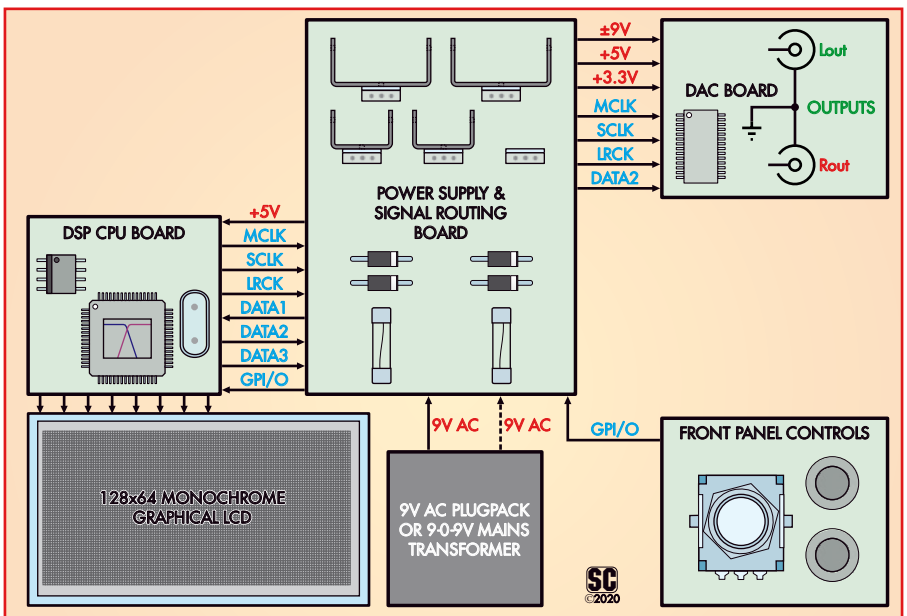
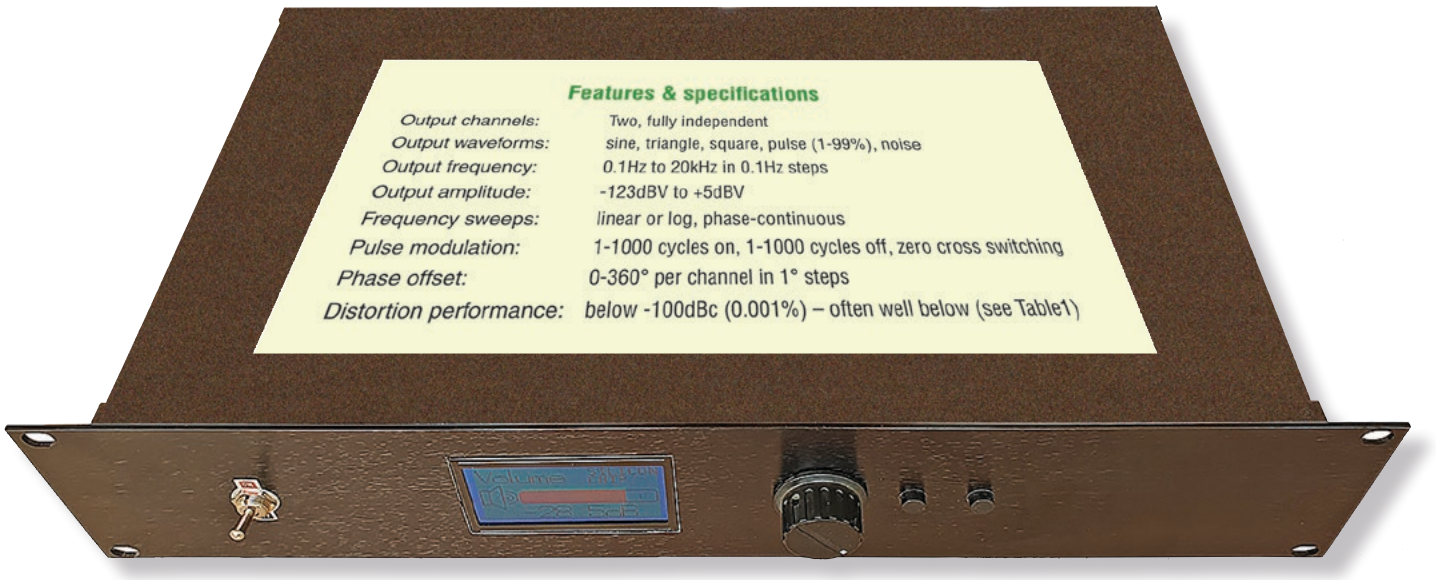


Fig.1: the powerful 32-bit PIC processor on the DSP CPU board generates a stereo digital audio signal which is routed to the DAC board (at upper right) through the central power supply/signal routing module (upper centre). The DAC board (upper right) converts the audio data into two analog output signals. It's all controlled via an LCD screen, one knob and two buttons.



The May 2019 article gave an overview of the hardware and then described how all the separate boards worked, except for the CPU control board and the front panel controls.

Those were covered in June 2019, along with the PCB assembly details.

The July 2019 article gave programming and final assembly instructions, along with usage instructions that are not directly relevant to this project, as the software is different. However, if you have seen that, the user interface of this new software will be familiar to you.

Those were long and detailed articles so we won't reproduce all that information here.

We'll just give a quick overview of how the hardware works and then jump into describing the new software.

The hardware

There are four PCBs involved in this project, and the basic arrangement is shown in Fig.1: first is the CPU board which hosts the powerful PIC-32MZ2048 32-bit processor, a couple of regulators and a crystal to provide an accurate clock source.

This connects to a power supply and signal routing board which derives the DC supplies required to power the various other boards from a 9V AC plugpack or transformer.

Digital audio signals from the CPU are routed through this central board to a stereo DAC board which provides the two analog outputs via onboard RCA connectors.

The fourth board is a front panel control board with a rotary encoder (which may have an integral pushbutton) and one or two separate buttons. The CPU board drives the graphical LCD module directly.

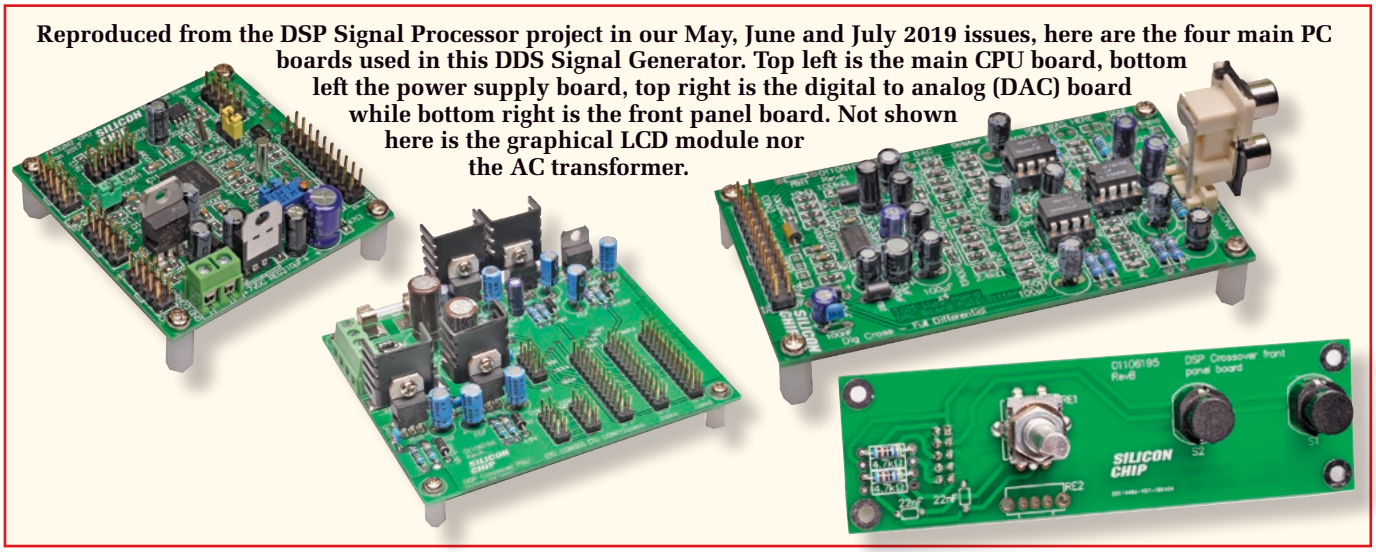
Fig.1 only differs from Fig.3 on page 28 of the May 2019 issue, which shows the DSP Active Crossover/Parametric Equaliser configuration, in that we've removed the unnecessary ADC input board and the second stereo DAC output board. Otherwise construction is identical.

Principle of operation

In case you aren't already familiar with how a DDS works, we'll give a quick description and describe why they are so much more useful than basic oscillators. We had a detailed description of DDS operation on pages 23 & 24 of the April 2017 issue, in an article on modules based around the AD9833 DDS chip. If you have that issue, you may wish to (re-)read that article now.

Direct digital synthesis is a process for mathematically generating a waveform. The system comprises a clock source, a 'phase accumulator' and a lookup table that determines what analog output is generated at any given time. Fig.2 shows this configuration.

The clock source runs at a high frequency compared to the output waveform frequency; the clock frequency is often hundreds of times more than the



Reproduced from the DSP Signal Processor project in our May, June and July 2019 issues, here are the four main PC boards used in this DDS Signal Generator. Top left is the main CPU board, bottom left the power supply board, top right is the digital to analog (DAC) board while bottom right is the front panel board. Not shown here is the graphical LCD module nor the AC transformer.

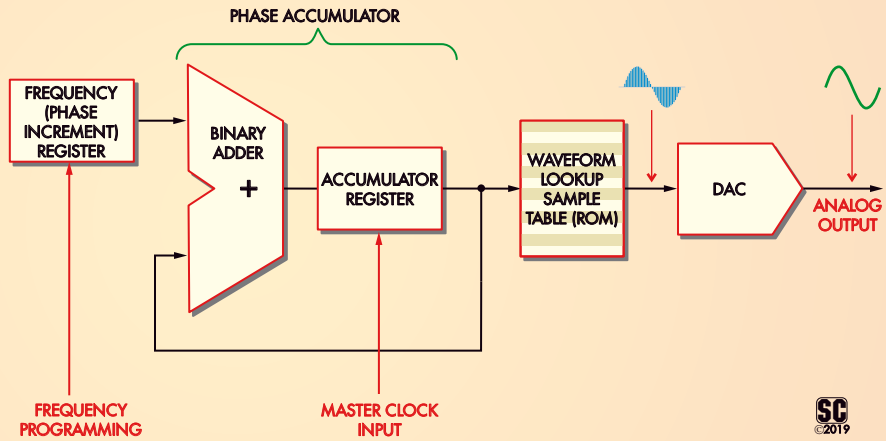


Fig.2: the basic configuration of a direct digital synthesiser. The phase increment value (chosen for a specific output frequency) is added to the phase accumulator on each pulse from the clock source. The accumulator is then used to index a waveform lookup table, and the values looked up feed the DAC to producing a varying analog waveform at the output. Its shape is determined by the values in the lookup table.

waveform frequency. On each clock cycle, the DDS system adds the phase increment to the phase accumulator. The lookup table can hold any waveform, though conventionally it would be a sinewave or something similar.

As the content of the phase accumulator increases in value, the system 'steps through' the table, feeding subsequent values in this table through to the output, reconstructing the waveform stored within. The rate at which it steps through the table (determined by the phase increment) determines the frequency of this reconstructed waveform, ie, how many times it runs through the table each second.

For example, if the clock frequency is 48kHz and you want an output frequency of 4800Hz, then you would need to produce one full sinewave every 10 clock cycles. Thus the phase increment needs to be 1/10th of the maximum phase value (equivalent to 36°).

If instead, you want to produce a 1Hz output with the same clock frequency, the phase increment needs to be 1/48,000th of the size of the phase accumulator (0.0075°). You can see then that for good low-frequency performance, a high-resolution phase accumulator is desirable. The PIC Microcontroller has a natural word size of 32 bits. That's suitable for moderate clock frequencies (up to a few MHz).

Also, to make the software simple, you want to use numbers in the DDS that the processor can divide easily. Using 32 bits makes things easy, since dividing by powers-of-two is very easy and fast (it can be done with simple shift/bit masking operations).

So we have a phase accumulator that is used to look up the waveform. For high precision, we need the sinewave to be very accurate. The limited memory in the PIC makes this a little tricky, as the largest practical sinewave table is around 4096 entries long. That's quite good, but not good enough for extremely low distortion.

The configuration of a basic DDS using a 4096-entry (12-bit) table is shown in Fig.3. This works reasonably well but, as shown, there are rounding errors in the values produced which makes it not quite a pure sinewave.

To solve this, we use the top 12 bits of the 32-bit accumulator to look up values from the 4096 entry (2^{12}) table, with the remaining (bottom) 20 bits determining where the current point in the waveform falls *between* samples in the table. This value can then be used to linearly interpolate between the two nearest (adjacent) entries in the table, to emulate having a much larger table.

The error in the resultant sinewave is vanishingly small. So the DDS configuration is now as shown in Fig.4. It's just a little more complicated than the one shown in Fig.3 and the extra calculations are well within the capabilities of a PIC32. The result is now

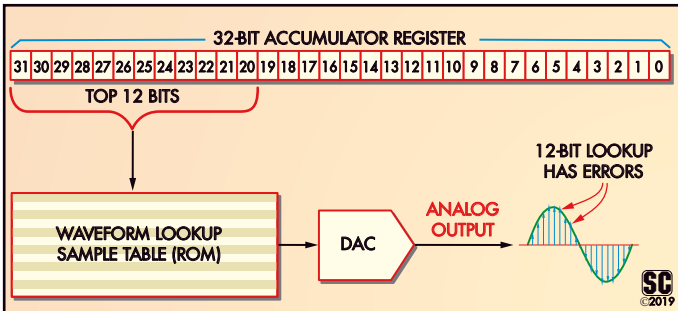


Fig.3: here's how the contents of a 32-bit accumulator are used to index values in a 4096-entry (12-bit) lookup table. The bottom 20 bits are ignored (although they are still needed to achieve the correct output frequency) while the top 12 bits are used directly as the table index value. This results in an output waveform with an accurate frequency but varying amplitude errors, leading to increased (but not necessarily excessive) waveform distortion.

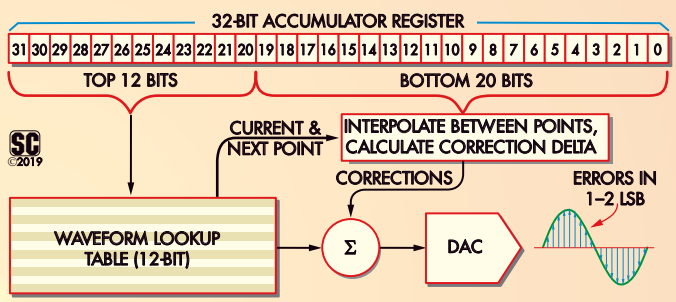


Fig.4: by adding a little extra complexity to the table lookup scheme, we can dramatically reduce the waveform distortion. The table lookup now retrieves the current and next values, and the bottom 20 bits of the accumulator are no longer ignored. Instead, they are multiplied with the difference between the two values from the table, to produce an error correction term which means that the values fed to the DAC are linearly interpolated between the table values. This reduces the instantaneous output amplitude errors to a tiny fraction of full-scale, and they essentially become negligible.

very close to a perfect sinewave at just about any frequency.

The DDS needs to do all these sums and interpolations 48,000 times a second. But this is what computer chips are really good at. The net result is that the data fed into the DAC is correct to within 1-2 least significant bits.

Without interpolation, spurs in the output are between -70dBc and -80dBc, which makes sense as the errors are about 1/4096 of the full amplitude (20 x $\log_{10}$ (4096) = 72dB). Fig.5 shows the frequency spectrum of the output without interpolation, and Fig.6 shows the same output with interpolation.

The DDS running without interpolation gives a creditable performance; those spurs (they are not harmonics) at -75dBc are around 0.02% of the waveform amplitude. For the version with interpolation, the largest signal by far is the hum being picked up by the test set at -100dBc or about 0.001% of the amplitude, with the distortion products being 1/10th of this or around 0.0001%!

So with interpolation, we can get a result close to the limits of the DAC's performance.

Along with getting the correct amplitude values to the DAC, we need to send them at the right time. Jitter in the clock which determines when data is sent to the DAC can also distort the resulting waveform. So we set up the PIC32MZ and the SPI port going to the DAC so that the notional DAC clock is an integral fraction of the PIC32MZ system clock of 252MHz.

By choosing the DAC clock this way, there is no jitter or error on the timing of the DAC signals. This is critical in all the clocks, phases and precisions being controlled. It does result in a rather unusual DAC clock rate of 49,218 samples per second, but that doesn't really matter, except to slightly complicate our phase increment calculations.

The PIC32MZ can generate a very accurate 48,000kHz clock, but this uses a 'trim' on the DAC clock, which introduces jitter. We don't want that!

This table shows how the Signal Generator's sinewave performance varies with the output frequency and amplitude. You can see that the harmonics are very low, only increasing to a significant level when the output amplitude is below -20dBV.

Frequency	Amplitude (on DDS)	Amplitude (Measured)	Harmonics
110Hz	0dBV	0dBV	-105dBc (0.00056%)
110Hz	-3dBV	-3dBV	-116dBc (0.00016%)
1kHz	0dBV	0dBV	-104dBc (0.00063%)
1kHz	-6dBV	-6dBV	-113dBc (0.00022%)
1kHz	-10dBV	-10dBV	-109dBc (0.00035%)
1kHz	-20dBV	-20dBV	-103dBc (0.00071%)
1kHz	-30dBV	-30dBV	Approx -90dBc (0.003%)
1kHz	-50dBV	-50dBV	Approx -70dBc (0.03%)
1kHz	-80dBV	-80dBV	Approx -40dBc (1%)
2kHz	-3dBV	-3dBV	-116dBc (0.00016%)
5kHz	-3dBV	-3.1dBV	-116dBc (0.00016%)

Table 1 - measured performance with various sinewaves

Construction & programming

As mentioned earlier, the construction steps were detailed in the June & July 2019 articles on the DSP Active Crossover. My only suggested change, other than leaving out the redundant second DAC board and the ADC board, is that you may wish to mount the DAC board with its two output RCA sockets coming through the front panel, rather than the rear. That makes it more practical to use as a test instrument.

The software for this project, both as a compiled HEX file and an MPLAB X IDE C project, can be downloaded from the Silicon Chip website. You can then upload that HEX file to the PIC32MZ chip using the procedure described on pages 87-88 of the July 2019 issue.

You can use that same procedure to reflash a DSP Active Crossover so that it can perform the DDS role outlined here. There is absolutely nothing stopping you from flashing it back to the Active Crossover / Parametric Equaliser firmware when you have finished with that.

Remember that these chips eventually *do* wear out if you keep reflashing them, but given the PIC32MZ's specification of a minimum of 10,000 erase cycles, you're unlikely to wear it out this way.

User interface

When the unit is first powered on, you can cycle through the following four main screens using the rotary encoder:

- Load settings
- Save settings
- Channel 1 options
- Channel 2 options

Once you've selected one of the channels, by pressing the Select button or pushing the rotary encoder, it jumps to the output frequency setting screen (see Screen 1).

Turning the dial does the obvious thing, ie, increments and decrements

Screen 1: this screen lets you set the output frequency of the currently selected channel.

Screen 2: here, you can change the size of the frequency steps in the previous screen.

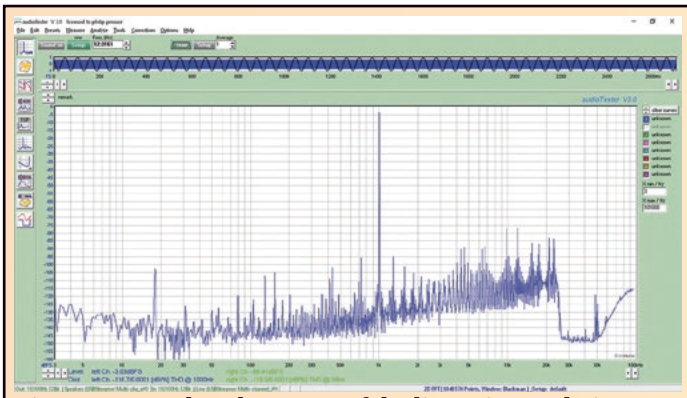


Fig.5: to prove the advantages of the linear interpolation scheme, here is a spectrum analysis (FFT) of the output of the unit using just the basic 12-bit lookup scheme shown in Fig.3 (ie, no linear interpolation). You can see that there are a large number of spurious signals at various frequencies present in the waveform, although most of them are not harmonics of the output sinewave. This leads to a distortion level of around 0.02% (the level shown in the screengrab is not accurate).

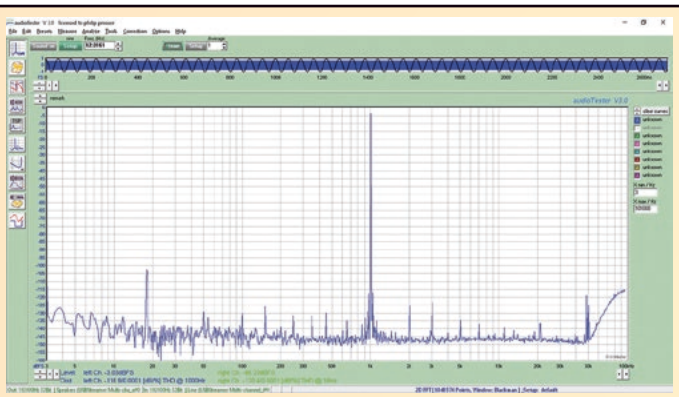


Fig.6: this is a spectrum analysis of exactly the same signal as in Fig.5, but this time, with linear interpolation. You can see that there is a lot less noise in the signal, and the actual harmonics are now visible, along with some mains interference at 50Hz, 100Hz and related frequencies.

the frequency. The Exit button returns to the channel selection screen, and the Enter button advances to the next menu page.

As you change frequency, if you turn the dial continuously for about a second or so, the step size increases by a factor of 10, allowing you to use both fine and larger frequency steps from the one screen.

This is one of a few data entry screens that are 'sticky'; if you leave the controls untouched for a long while, the interface will remain on this screen, rather than reverting to the initial screen. The reasoning here is that if you are fine-tuning a speaker or a filter, you may need to make the occasional frequency adjustment and it would be annoying to have to go through the menus again each time.

If you press the select button/knob on the frequency setting screen, it takes you to a step adjustment screen (Screen 2), which lets you increase the base step size from 0.1Hz to 1Hz, 10Hz or 100Hz. This may be useful if you are making lots of rapid changes to the output frequency and don't need to be extremely precise.

The behaviour of the Exit button on this screen is different; exiting from this screen takes you back to the previous frequency setting screen, rather than the initial screen. Pressing Enter on this screen takes you to the output level adjustment screen (Screen 3).

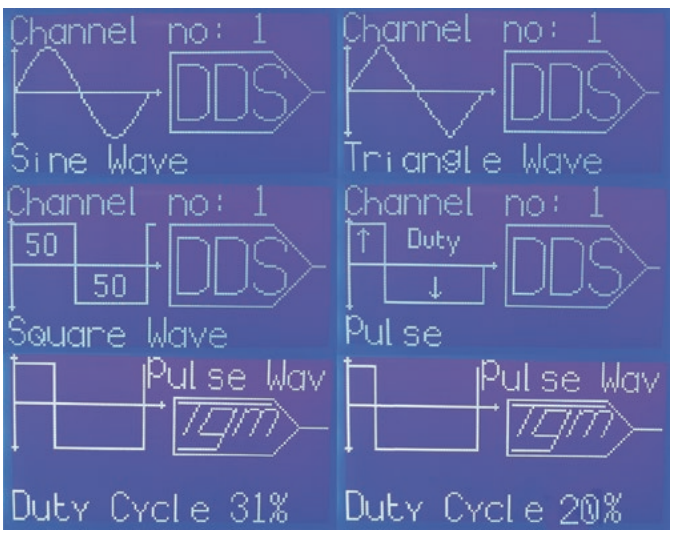
Screen 3: the next screen lets you set the output level from -123dBV (0.7mV RMS) to +5dBV (1.78V RMS) in steps of 0.1dBV.

The output level is set in dB Volts (dBV). 0dBV = 1V RMS, -20dBV = 0.1V RMS, -40dBV = 0.01V RMS etc. The resolution is 0.1dB, which is quite a small step. As you rotate the control, if you rotate for more than a second or so, the increment size increases (as with the frequency adjustment), allowing you to make larger changes in output level reasonably quickly.

The output range goes from +5dB Volts (1.78V RMS) to -123dB Volts (0.7mV RMS), but note that -123dB Volts is essentially the noise floor of the DAC. The Exit button takes you back to the main screen, while pressing Enter takes you to the waveform selection menu (Screen 4).

On this page, rotating the encoder cycles through the various waveform types, including Sine, Triangle, Square, Pulse and Noise. If you select Pulse, a secondary menu pops up, allowing you to set the duty cycle in 1% increments.

Having chosen a new waveform, press Select to reprogram the waveform lookup table and change the output signal. This also takes you to the next screen, which is the sweep



Screen 4: as you rotate the knob on the waveform selection screen, you can select Sine, Triangle, Square or Pulse. For Pulse, you can set the duty cycle from 1% to 99%.

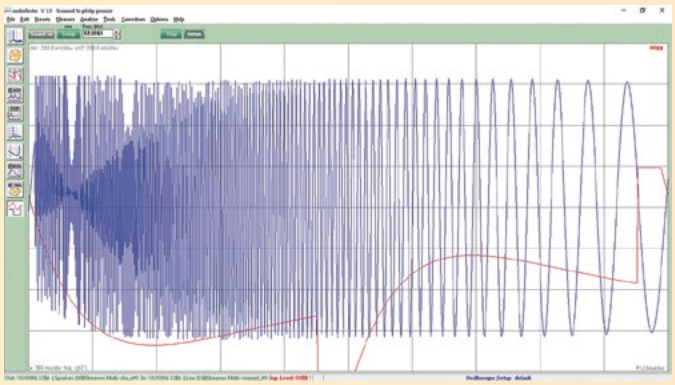


Fig.7: an example output waveform from the unit in sweep mode (blue trace). It is sweeping from a high frequency down to a low frequency.

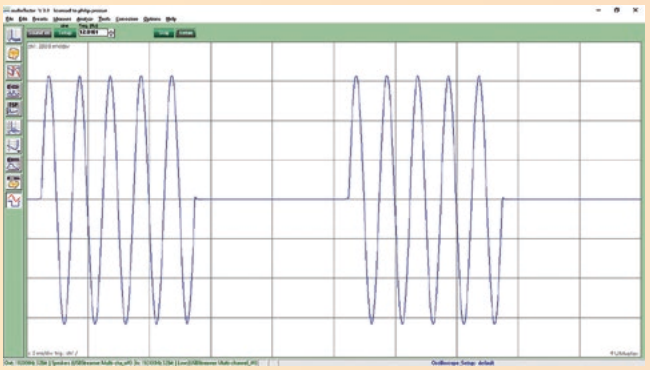


Fig.8: a sample pulse train output, with five pulses on (at 1kHz), followed by five pulses off. This sequence then repeats endlessly.

selection menu (Screen 5). As usual, pressing Exit will take you back to the channel selection menu.

Sweeping and pulses

Sweep options include None, Linear and Log (logarithmic). Logarithmic sweeps (actually exponential) are very useful for wideband tests and audio testing. Your ear is much better 'tuned' to a logarithmic sweep than linear, giving the sense of a constant rate of pitch increase or decrease.

The frequency at which the sweep starts is the frequency set on the main frequency menu. Having chosen the sweep type, you can then adjust the sweep end frequency. This may be above or below this start frequency.

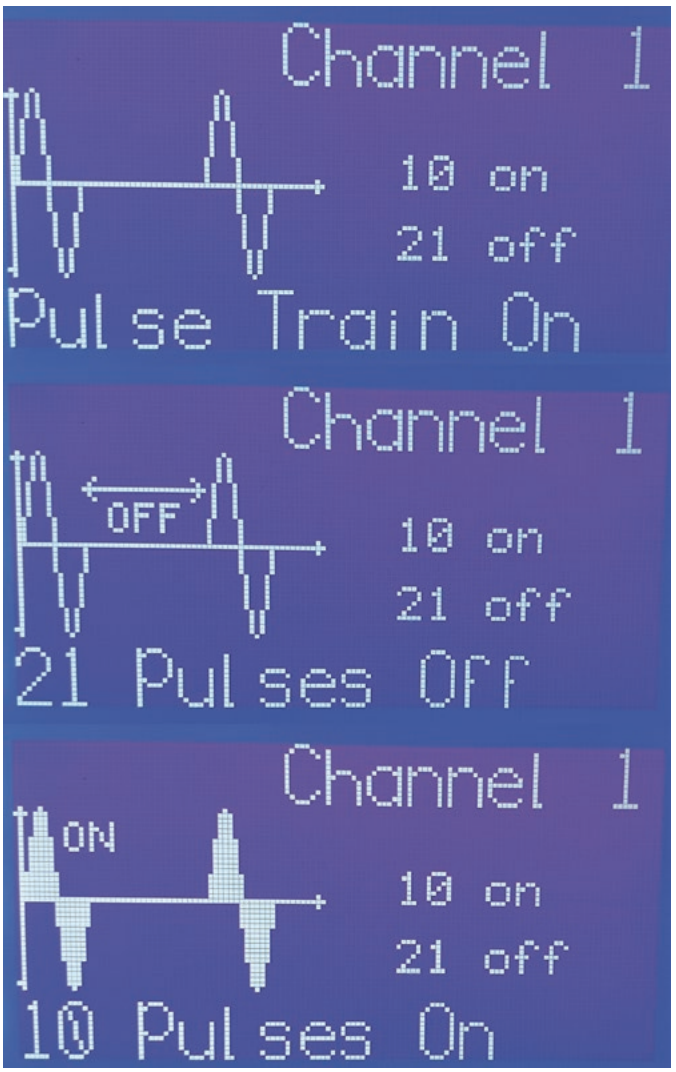
This allows sweeps upward and downward without complicating the interface.

A sweep is implemented by the DDS frequency being changed in 4096 steps throughout the sweep. Without getting into too much detail, the sweep involved repeatedly changing the phase increment value.

This has the benefit of the output phase from the DDS always being continuous (ie, no phase 'jumps'). A sample of the sinewave output in sweep mode is shown in Fig.7.

Having adjusted the end frequency and pressed Select, you can then adjust the sweep time from 0.1 seconds to 60 seconds in 0.1-second steps using the knob. Press Select

Screen 5: you can disable sweeping, or have a linear or log (exponential) sweep. If enabled, you set the end frequency and sweep time in the next couple of screens (the start frequency is the channel's set frequency). It can sweep up or down.



Screen 6: the unit can be set to produce a continuous waveform or a pulse train. With a pulse train, you can have 1-1000 pulses followed by 1-1000 gaps. Each pulse or gap is the set waveform period (the reciprocal of the frequency), eg, 1ms for 1kHz, 10ms for 100Hz etc.

again to move onto the next screen, or Exit to go back to the main selection screen.

The next screen configures pulse train mode (Screen 6). The pulse train can be set to off or on. If you turn pulses on, you need to set the number of pulses on and off for the train. Both can have values of 1-1000 pulses. So with a 1kHz tone, you could have a single 1kHz cycle followed by one second of silence.

As noted earlier, this is very handy for high power amplifier testing (especially stability testing into very nasty or low impedances), and for testing loudspeaker driver phase centres. Fig.8 shows an example of the output using the pulse mode, with a 1kHz waveform set for five pulses on and five pulses off.

Pressing Select after the last pulse mode screen takes you to the phase shift screen (Screen 7), which allows you to change the relative phase between the two output channels. This only really makes sense where both channels are set to identical frequencies. This is particularly handy when you want to use one channel to trigger an oscilloscope. The phase shift lets you move the waveform of the second channel on the oscilloscope screen.

Conclusion

All the PCBs to build this project are already available from the **Silicon Chip Online Shop**. As we mentioned earlier, they were also used in the DSP Active Crossover/Parametric Equaliser project. They are coded 01106192-01106196. We can also supply a PIC32MZ microcontroller pre-programmed with the software for either project.

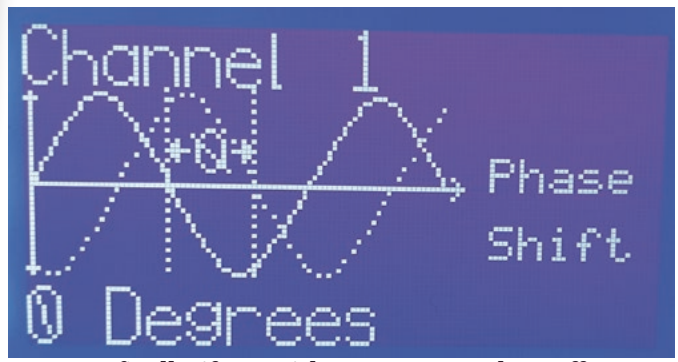
Many of the parts are SMDs, some of them fine-pitched, so this is not an ideal project for beginners.

But you will get a test instrument with excellent performance, that can carry out many important audio tests, especially for hifi gear.

You could even combine this Signal Generator with a high-quality ADC connected to a computer, feeding into spectrum analysis software, as an audio analyser capable of measuring distortion in devices like amplifiers, preamplifiers and filters, down to very low levels.

It can also be used in conjunction with our High-Resolution Audio Millivoltmeter project, published in the October 2019 issue (siliconchip.com.au/Article/12018), for making signal-to-noise ratio measurements.

That would be a great way to make and save frequency response plots, with the Signal Generator in sweep mode and the Millivoltmeter connected to a USB port, for logging the results, which could then be fed to a plotting program.



Screen 7: finally, if you wish you can set a phase offset between the two channels, although this only makes sense if they are set to the same frequency. SC