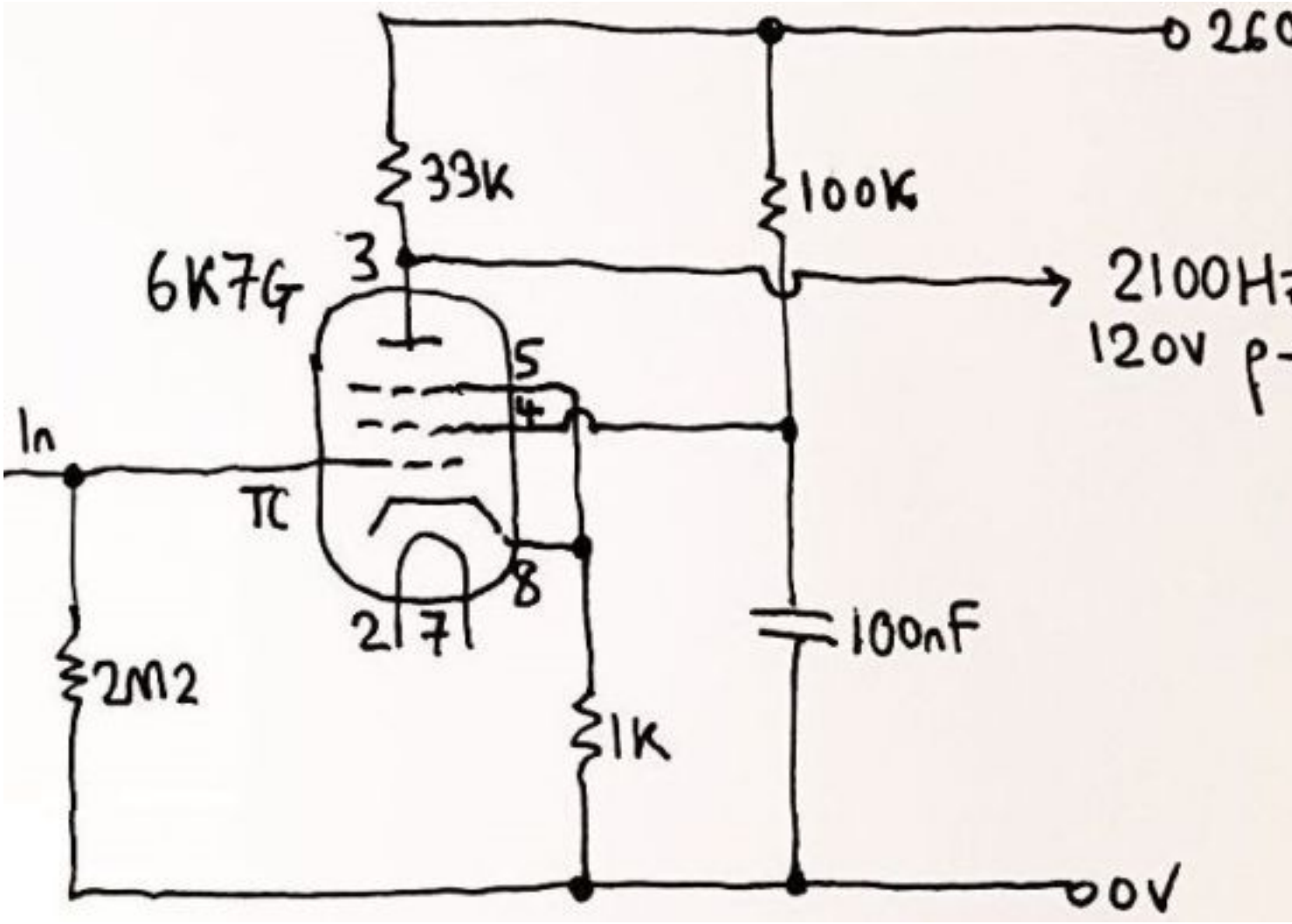
This is a phantastron divider based on the HP522 frequency counter circuit diagram . The input is a 2100Hz 15V peak-peak signal from my
2.1kHz oscillator project
. Please take a look at the
crystal oscillator page
! The
HP522 manual
has a good explanation of how a phantastron divider works. So have a read of that, for the theory. My project has two stages. One is divide-by-21, one is divide-by-10. The resulting timebase outputs are 100Hz and 10Hz.

Buffer stage

Firstly, I discovered that 15V peak-peak from the crystal oscillator is NOT enough to clock the phantastron circuit reliably. I did not wish to increase the supply voltage of the
crystal oscillator
to get more output, because I do not want to risk any damage to the crystal (e.g. fracture) by over-stressing it. The practical solution was just to build a simple amplifier stage. Initially I used a CV136 pentode but I had a couple of 6K7G octals in the junk box and thought they look cool, so I used one of those. I know the big octal looks out of place next to the mini B7G 5915 pentagrids and the tiny 6AL5... but I like it and I'm sticking with it. It's a crazy project anyway and nothing has to make sense, by definition. The buffer stage output is 120V peak-peak. Which is PLENTY to operate the phantastron.

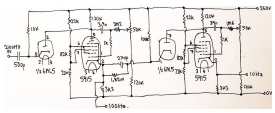


Phantastron stages

I used the 5915 pentagrid, and 6AL5 dual diode, the same as HP used in their HP522 frequency counter
. My supply voltage measures about 240V so is a bit higher than their 200V. I did not have much trouble getting this working, first starting with the divide-by-21 stage. Initially the division ratio adjustment potentiometer did not make much difference, I traced that eventually to a dry joint ground connection at the bottom of the potential divider's 120K resistor. After that, I could change the division ratio from 19 to 25.

This is a photo of the divide-by-21 stage during development (click for larger image).

{gallery}td/phantastron/testing{/gallery}



The 10Hz and 100Hz outputs are about 30V peak-peak. The accurate edge is the negative (falling) edge of the waveform.

I built a nice little chassis for the project, from single-sided FR4 PCB laminate material soldered together at the seams inside. The two 50K potentiometer adjustments are available on the top side of the chassis. The first division stage can be set to divide by a ratio from 19 to 25; the second stage can be set to divide by a ratio from 9 to 12. The top cap (grid 1) connection on the 6K7G is made with a clip that I made from a piece of tin cut from an ex- food tin can.

{gallery}td/phantastron/photos{/gallery}

Here are the oscilloscope screen captures. In each case the input to the division stage is the top trace, the phantastron divider output is the bottom trace.

{gallery}td/phantastron/scope{/gallery}

10Hz Precision test

This is a test designed to check that the 10Hz output is precise, without any jitter or noise (such as 50Hz mains power hum problems). The test does not investigate the accuracy of the 2100Hz crystal oscillator (which is anyway adjustable) or the temperature drift properties of the oscillator etc. The strategy is to use an Arduino Uno to count a test signal, using the 10Hz signal as a timebase gate.

The circuit diagram and photograph of the test set-up is below (click for larger versions). The 10Hz signal output from the phantastron divider is at 30V peak-peak and we need a 5V

peak-peak squarewave. First I used a 10K:47K potential divider to reduce the signal to about 5V peak-peak. Next, half an LM4562 dual op-amp as a comparator, with its threshold set to 2.5V by the two 10K resistors. A 10K feedback resistor to the threshold voltage gives the comparator some hysteresis to prevent multiple false triggering at the the transition point. This LM4562 output in this circuit seemed to prefer the positive rail more than the negative rail, having some offset. So the 1N4001 diode shifts the waveform down a bit (maybe about half a volt). The first half of the 74LS74 dual D-type flip flop is a simple divide-by-2 circuit that results in a precise 50% duty cycle 5Hz squarewave at the output. The second half of the 74LS74 divides a 2MHz test signal by two. The 2MHz test signal was generated by an Ultimate3S transmitter kit sending a message in FSKCW at 2MHz with 0Hz FSK setting (meaning, just an unmodulated 2MHz signal). The /PR (preset) input of the 2MHz divide-by-2 is connected to the 5Hz signal so that when the signal is low, the flip flop is held in its pre-set (1) state, effectively gating the count. The result is a stream of bursts of 1MHz signal, each burst is 0.1 seconds long, and there are 5 bursts per second.

It is reasonably straightforward to get an Arduino Uno to count this signal. The 1MHz bursts are connected to Arduino Uno pin 5, which is connected to the ATmega328's 16-bit Timer/Counter1 input. The gating signal is fed to Arduino pin 7 so that the Arduino can sense the transition from high to low, then it outputs the count to the serial port and resets it to zero ready for the next 1MHz pulse burst. I also used the Arduino board to supply 5V power to the comparator and 74LS74 circuit.

{gallery}td/phantastron/10Hztest{/gallery}

This is the simple Arduino sketch used:

```
  int prev;  int p;  int q;    void setup() {        // Open serial communications and wait for port to open:    Serial.begin(9600);    while (!Serial) {      ; // wait for serial port to connect. Needed for native USB port only    }      pinMode(5, INPUT);    pinMode(7, INPUT);    TCCR1A = 0;    TCCR1B = 7;    TIMSK1 = 0;  }    void loop() { // run over and over      if ((TCNT1 < 32768) && q)    {      q = 0;      p++;    }    else if (TCNT1 > 32768) q = 1;      if (digitalRead(7) != prev)    {      prev = digitalRead(7);      if (!prev)      {        Serial.println(p * 65536 + TCNT1);        p = 0;        q = 0;        TCNT1 = 0;      }    }  }
```

On the serial monitor, this resulted in a stream of 100,022 or 100,023 counts being recorded. Note that the least significant digit variation is common to any simple frequency count, and simply due to the non-integral count; so that the final least significant pulse edge sometimes falls in one count bucket, and sometimes in the next. In other words, it is completely normal.

100023

100022
100022
100023
100022
100022
100023
100022
100022
100023
100022
100022
100022
100022

I let the test run for more than 20 minutes, the count was always 100,022 or 100,023. This indicates the 10Hz signal's falling edge is very stable, with no problems from hum or noise etc. I am very pleased with the result.