CNT-104S

Synchronizing 2x Multi-channel Frequency Analyzer CNT-104S for phase comparison of up to 8x 1-pps signals

APPLICATION NOTE

Author: Staffan Johansson

Background

CNT-104S is a Multi-channel Frequency Analyzer / Time Interval Analyzer, capable of simultaneously measuring 4 parallel input signals. This multi-function feature can for example be used to phase-compare 4x 1-pps signals in synchronization hubs using the four parallel time-stamping engines running on the same time scale.

But if you would like to compare 8 different 1-pps signals, you cannot simply just put 2x CNT-104S on top of each other, or mount 2x CNT-104S side by side in a 19" 2U rack:

- With 1x CNT-104S you could phase-compare 4x 1-pps pulses
- With 2x CNT-104S you can connect 8x 1-pps pulses, but the timescale of the two CNT's will be independent, and not synchronized
- The procedure below will synchronize two CNT-104S to a common timescale with 8 inputs

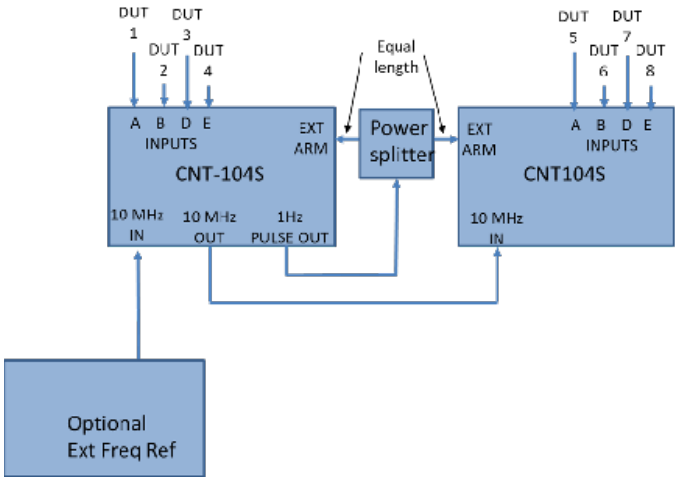

Figure 2. Set up for 8-channel phase comparison

Basic principles:

To generate a common timescale, we will use the "ARMING on block" feature in CNT-104S. When set, the Arming feature blocks any input trigger event until the Arming event, when the timescale is set to zero. The two CNT-104S are armed at exactly the same moment, resetting the timescale to zero in both units at the same time.

The 8 DUTs are thereafter timestamped relative to the arming timestamp.

Subsequent Arming trigger events will be neglected when selecting "Arming on block"

Downloading the measurement data from both units will give 8 timestamps with the same timescale, with an uncertainty of <3ns, due to delay differences in the arming circuitry between the two units, and residual delay difference in the two cables that should have equal length.

When setting up this 8-channel parallel timestamping from a controller using SCPI commands, the preferred measurement function is *"Function=TIMESTAMPS A, B, D, E"* which simply returns the timestamp of each trigger event in each channel.

When using the GUI or Web Interface, the preferred measurement function is *"Time Interval Accumulated A, B, D, E"* with file save of measurement data and export to MS Excel for postprocessing.

Although the application note describes how to measure phase between 1-PPS pulses, the same HW configuration can be used to phase compare RF signals to 300 MHz. For input frequencies <20MHz, *every* trigger event can be timestamped

For input frequencies >20MHz, phase samples will be made at set sample interval down to 50ns

Detailed set-up

- Connect as shown in figure 2.
- Select Recall Default in both units (SETTINGS→USER OPTIONS→RECALL DEFAULTS→YES)
- Set-up both Frequency Analyzers for the synchronization measurement.
  - Set Manual trigger levels at 50% of the pulse amplitude on all inputs (SETTINGS→INPUTS→X→TRIGGER LEVEL = MANUAL, ABSOLUTE TRIGGER LEVEL X = <Desired Trigger Level Value>, where X is particular input)
  - Set 50 ohm input impedance for the 1-pps pulses (SETTINGS→INPUTS→X→MPEDANCE = 50 OHM, where X is particular input)
  - Set DC coupling for the 1-pps pulses (SETTING→INPUTS→X→COUPLING = DC, where X is particular input)
  - Set Measurement function Time Interval Accumulated A,B,D,E (SETTINGS→MEASUREMENT→FUNCTION→ TIME/PHASE->TIME INTERVAL ACC. A, B, D, E)
  - Set Sample Interval =0 (SETTINGS→MEASUREMENT→SAMPLE INTERVAL = 0)
  - Set desired Sample Count (SETTINGS→MEASUREMENT→SAMPLE COUNT = <Desired Sample Count Value>). E.g. 86401 samples means a 1 day recording, 604801 means 1 week recording, and 2592001 samples means 1 month (30 days) recording.
- Set up block arming.(SETTINGS→ARMING→SOURCE = EA, ARM ON = Block)
- Set up Pulse output in CNT-104S for 1s pulses. (SETTINGS→PULSE OUTPUT ON, 1s period).
- Enter HOLD mode.
- Start measurement in SINGLE mode (up to 10 million 1-pps values/channel can be sampled for 115 days)
- The relative timescale will now start from 0 in both counters simultaneously (Arming event)
- Switch OFF Pulse output after start of measurement (that does not affect the measurement, which will continue uninterrupted until end of sample counts)
- Save/stream the result file to e.g., MS Excel, and process the result:
- The result file has two columns; timestamp for start trigger on A, Time Interval result value, TI(A-X).
- The Arming event, the start of the time scale, has timestamp T0 = 0
- For every sample, DUT 1 has Timestamp T1. DUT 2, 3, and 4 has Timestamp T1+TI(A-B), T1+TI(A-D), T1+TI(A-E), where TI(x-y) is the Time Interval value
- DUT 5 has Timestamp T5. DUT 6, 7, and 8 has Timestamp T5+TI(A-B), T5+TI(A-D), T5+TI(A-E)

Uncertainty in timescale between CNT-104S units is <3ns (typically 1 ns). The time from input arming edge to "ready for trigger" in any of the units is <5ns, but the difference between units is < 3ns

(Uncertainty of individual channel offsets in each CNT-104S is <25ps rms, which is neglectable compared to the arming uncertainty)

Enhancements – real-time UTC timescale

Create a real-time time-scale by using an external 1-pps UTC reference signal to arm the measurement. Then all timestamps will be referenced to UTC, instead of the 1pps pulse output from Frequency Analyzer number one.

The timescale uncertainy to UTC will be:

(Uncertainty of external 1-PPS) ± 3 ns

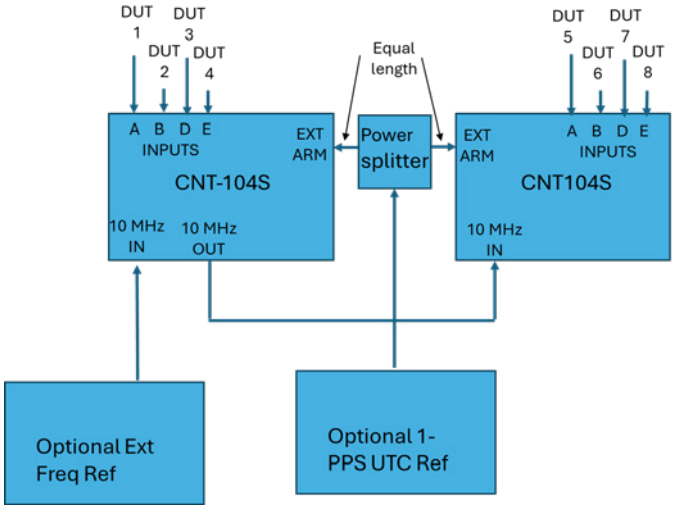

Figure 3. 8-channel parallel timestamping with a real-time timescale referenced to UTC

Enhancements – 6-channel timescale with 50ps (typ.) uncertainty

The time scale uncertainty between the two CNT-104S in previous 8-channel example is <3ns. However, if we use one of the A,B,D,E inputs in the CNT-104S units for arming instead of the Ext Arm input, we can drastically reduce the timescale uncertainty from <3ns to typically 50 ps. This is done at the expense of number of DUTs, now maximum 6 DUTs instead of 8.

The HW setup is quite similar to the previous setup, except that in the example below, we use input E in counter/analayzer #1, and input A in counter/analyzer #2 as arming inputs. Then the time skew between Arming to measurement inputs is reduced to <25ps rms in each CNT-104S

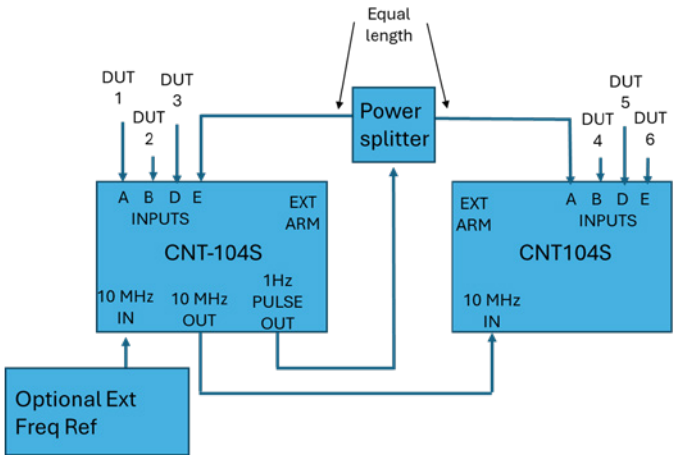

Figure 4. Setup for 6-channel parallel timestamping with typ. 50 ps uncertainty in the timescale, vs. typ. 1ns for 8 channels

Conclusion

The CNT-104S Multi-Channel Frequency Analyzer, with or without post-processing of the data files in e.g. MS Excel, Matlab, or Stable32, can be used for several measurements common to synchronization tests

- Phase comparisons of 4 or 8 sync sources
- Verification, calibration and adjustment of period/pulse width (1 PPS)
- Wander parameters (TIE)
- Short-term-stability test (ADEV)
- Verification of frequency/period stability (finding micro-glitches)
- PLL parameter testing

CNT-104S Timer/Counter/Analyzer features

- Multi-Channel Frequency Analyzer for parallel testing of 4 DUTs
- Ultra-high resolution and speed
- Built-in in TIE and ADEV measurements
- And - last but not least – the CNT-104S provides a very cost-effective solution for parallel test, replacing up to 4 traditional timer/counters

Programming example (Python script)

```
# This example demonstartes how to timestamp up to 8 x 1 PPS signals in parallel
# using 2 synchronized CNT-104S

# The setup is:
#
# Common 10 MHz reference -> CNT-104S #1 Ext Ref IN
#                         -> CNT-104S #2 Ext Ref IN
#
# CNT-104S #1 Pulse Output -> power splitter -> CNT-104S #1 Ext Arm IN
#                                            -> CNT-104S #2 Ext Arm IN
#
# DUT1 -> CNT-104S #1 input A
# DUT2 -> CNT-104S #1 input B
# DUT3 -> CNT-104S #1 input D
# DUT4 -> CNT-104S #1 input E
# DUT5 -> CNT-104S #2 input A
# DUT6 -> CNT-104S #2 input B
# DUT7 -> CNT-104S #2 input D
# DUT8 -> CNT-104S #2 input E

import pyvisa as visa
from time import sleep

# helpers

def cnt_connect_and_reset(resource_manager, ipv4: str):
    cnt = resource_manager.open_resource(f'TCPIP::{ipv4}::hislip0')

    cnt_id = cnt.query('*IDN?')
    print(f'Connected to: {cnt_id}')

    # reset to the known default state
    cnt.write('*RST;*CLS')

    return cnt

def cnt_check_errors(cnt, operation_name: str):
    error = cnt.query(':SYST:ERR?')
    error_code = int(error.split(',')[0])
    if error_code != 0:
        raise RuntimeError(f'{operation_name} failed: {error}')

def cnt_configure(cnt, configuration: str):
    cnt.write(f':SYST:CONF "{configuration}"')

    cnt_check_errors(cnt, 'Configuration')

def cnt_set_format(cnt, configuration: str):
    cnt.write(f':SYST:CONF "{configuration}"')
```

```
    cnt_check_errors(cnt, 'Configuration')

def cnt_start_measurement(cnt):
    # enable operation complete (OPC) bit in the status register
    cnt.write(f'*ESE 1')

    # start the measurement
    cnt.write(':INIT;*OPC')

def cnt_operation_complete(cnts):
    complete = True
    for cnt in cnts:
        stb = cnt.read_stb()
        if (stb & 32) == 0:
            complete = False

    return complete

def cnt_extract_timestamps_from_ascii(response: str):
    # ASCii format with timestamps is:
    # <VAL0>,<TS0>,<VAL1>,<TS1>,...
    # so all odd elements in the list are timestamps
    float_array = response.split(',')
    timestamps = float_array[1::2]
    return [float(t) for t in timestamps]

def cnt_cleanup(cnts):
    for cnt in cnts:
        cnt.close()

# main script

if __name__ == '__main__':

    # Create VISA resource manager. In some cases you may need to specify dll/so
    # file of VISA implementation, for example:
    # rm = visa.ResourceManager('librsvisa.so')
    rm = visa.ResourceManager()

    # CNT-104S that will provide common arming via Pulse Output
    cnt_1 = cnt_connect_and_reset(resource_manager=rm, ipv4="10.40.15.23")
    # 2nd CNT-104S
    cnt_2 = cnt_connect_and_reset(resource_manager=rm, ipv4="10.40.15.26")

    print('Configuring measurement...')

    # configure common arming signal
    cnt_configure(
        cnt_1,
        'PulseOutputMode=PulseGenerator;'
        'PulseOutputPeriod=1.0;'
        'PulseOutputWidth=0.01'
    )

    cnts = [cnt_1, cnt_2]
    inputs_to_measure = ['A','B','D','E']
    sample_count = 100

    for cnt in cnts:
        # measurement configuration
        cnt_configure(
            cnt,
            f'Function=Timestamps {",".join(inputs_to_measure)};'
            f'SampleCount={sample_count};'
            'SampleInterval=0.0'
        )

        # arming configuration
        cnt_configure(cnt, f'StartArmingSource=EA; ArmOn=Block')

        # inputs configuration
```

```
    for input in inputs_to_measure:
        cnt_configure(
            cnt,
            f'TriggerMode{input}=Manual;'
            f'AbsoluteTriggerLevel{input}=0.3;'
            f'Coupling{input}=DC;'
            f'Impedance{input}=50 Ohm'
        )

    # set ASCii format (for example code simplicity) with timestamps
    # note: in real life binary form (PACKed) is preferable most of the time
    cnt.write(':FORM ASC;:FORM:TINF ON')

print('Measurement configuration completed')

# start measurement
for cnt in cnts:
    cnt_start_measurement(cnt)
print('Measurement started')

# prepare timestamp storage
series = []
for _ in range(0, len(cnts) * len(inputs_to_measure)):
    series.append([])

# fetch data in real time
measurement_complete = False
while True:
    if not measurement_complete:
        measurement_complete = cnt_operation_complete(cnts)

    # fetch all series one by one
    nothing_fetched = True
    dut_index = 0
    for cnt in cnts:
        for input in inputs_to_measure:
            samples=cnt.query(f':FETCH:ARR? MAX, {input}').strip()
            if len(samples) > 0:
                timestamps = cnt_extract_timestamps_from_ascii(samples)
                series[dut_index].extend(timestamps)

            dut_index += 1

    if nothing_fetched:
        if measurement_complete:
            print('Measurement completed')
            break;

        # pace FETCh queries if no data is ready yet
        sleep(1.0)


# print results
print('Results:')
separator = ', '

series_names = [f'DUT #{i + 1}' for i in range(0, len(series))]
header = separator.join(series_names)
print(header)

data = list(zip(*series))
for row in data:
    print(separator.join([f'{t:0.17g}' for t in row]))
```


© Pendulum Instruments 2025

20.10.2025 AN-04 rev.1

Specifications subject to change or improvements without notice.

4031.601.00461