HERA Memo #100*
Received 15 July 2021

Synchronizing the HERA Correlator to an LST Grid

July 15, 2021

by Aaron Parsons

1 Purpose

The purpose of this memo is to define the time grid in local sidereal time (LST) to which we will lock integrations from the HERA Correlator. In doing so, we also define the maximum integration time for the correlator employing baseline-dependent averaging (BDA). By locking integrations to an LST grid, we avoid having to interpolate between integrations when stacking or comparing days of observation.

This memo supports an upgrade to the HERA Correlator control between the H4C and H5C seasons. Prior to this memo, LST grid alignment was achieved by attempting to find a GPS-derived 1PPS (one pulse per second) close to an LST grid boundary and arming F-engines to reset their master counters (MCNTs) on that 1PPS. This system had the shortcomings that * the actual LST grid of the observations depended on run-time operation (i.e. it emerged from the start time rather than being pre-defined), * little attention was given to how evenly the chosen LST grid divided a sidereal day, and * there was no support for LST alignment under BDA.

The goal of the new system defined by this memo is to address these shortcomings.

2 Background

The HERA correlator consists of three major subsystems: the F-Engine responsible for synchronization and channelization, the X-Engine responsible for cross-correlation and accumulation, and the data recorder that writes accumulations and meta-data to files on disk.

F-Engine synchronization is achieved by distributing phase-aligned sample clocks and 1PPS signals to each HERA SNAP board using a network-based synchronization system known as White Rabbit. F-Engines re-synchronize on the next 1PPS signal when an 'arm' command is sent.

On the rising edge of the next 1PPS, they reset the MCNT counter which count the number of subsequent spectra. This counter is embedded in the packets output by the F-Engine and is used by the network receive buffer on the GPU-accelerated X-Engines to time-align spectra for cross-correlation.

After cross-correlation, visibility products are accumulated on the X-Engine for an integer number of MCNTs, as specified in configuration. We will call this parameter $MCNT\_PER\_INT$. In the current (non-BDA) correlator, all accumulation buffers use this same parameter. After transitioning to BDA, the correlator will support power of 2 subdivisions of this parameter for baselines, as appropriate for their length, up to 2^4.

*When we make reach 200, there will be cake.

For the purpose of BDA, we usually concern ourselves with the minimum number of MCNTs per integration, but for defining an LST grid for the correlator, we are actually more concerned with the maximum integration time, since this defines the coarsest grid for alignment. Thus, for this purpose of this memo, $MCNT\_PER\_INT$ is taken to be correspond to the longest integration time.

3 Architecture and Constraints

Under the proposed system, an absolute LST grid is pre-defined by dividing a sidereal day into an integer number of bins, each spanning a time of

$$t_{\text{grid}} \equiv t_{\text{sidereal}}/NGRID \tag{1}$$

while simultaneously satisfying

$$t_{\text{grid}} \approx MCNT\_PER\_INT \cdot NSAMP\_PER\_MCNT/f_{\text{clk}}, \tag{2}$$

where $NSAMP\_PER\_MCNT$ is the number of time samples used to compute one spectrum in the F-Engine (nominally 2^{14}) and f_{clk} is the sample clock (nominally 500 MHz).

Because t_{grid} as realized in the second equation above may only approximately divide t_{sidereal}, there will be drift of the correlator's integration boundaries with respect to our absolute LST grid. We will take the LST grid indices to be defined by an even division of $[0, 2\pi)$ into $NGRID$ bins:

$$LST\_GRID\_INDEX \equiv \text{floor}\left[NGRID \cdot \frac{LST}{2\pi}\right]. \tag{3}$$

Our goal is to find a value for $MCNT\_PER\_INT$ such the LST_GRID_INDEX can be calculated as accurately as possible using integer arithmetic on $MCNT$. We will do this by choosing an $MCNT\_PER\_INT$ that closely matches t_{grid}, and by calculating an offset of $MCNT$ relative to the edges of our LST bins. When the correlator is synchronized (each day, to avoid accumlating drift), any arbitrary 1PPS event can be used; the synchronization script records an MCNT_OFFSET that, when subtracted from the MCNT, aligns integrations to the LST grid defined above. Thus, LST_GRID_INDEX may be approximated inside the correlator as

$$LST\_GRID\_INDEX \approx \text{floor}\left[\frac{MCNT - MCNT\_OFFSET}{MCNT\_PER\_INT}\right] \ \%\ NGRID \tag{4}$$

3.1 Additional Constraints

Internal correlator buffering and our choices for BDA integration times present additional constraints on $NGRID$ and $MCNT\_PER\_INT$.

The number of spectra spanned by an LST grid cell ($MCNT\_PER\_INT$) should be divisible by 32, supporting grid subdivisions for BDA while ensuring alignment in LST.

Also, because correlator packets bundle 4 spectra together (and we are unsure if the X-engine can handle them separately), we require that $MCNT\_PER\_INT$ be additionally divisible by 4, which is to say that $2^5 \cdot 2^2 = 2^7$ must divide $MCNT\_PER\_INT$ evenly.

Finally, we expect the maximum integration time for BDA to be in the neighborhood of 16 s, but in order to have headroom to tweak this later, we will adopt a target value of

$$t_{\text{grid}} \approx 32\text{s}. \tag{5}$$

The cost of this headroom is just an added 16 s of wait-time for correlator synchronization, which should not matter.

4 Numerology

Assuming a sample clock of 500 MHz, a real-valued FFT input of 16384 samples, and correlator packets that bundle 4 spectra per packet, let us first compute the number of packets in a sidereal day, which we will assume defines the fundamental time resolution of correlator integrations.

[1]:
```
t_sidereal = 86164.0905 # s / sidereal day
f_clk = 500e6 # Hz
NSAMP = 2**14 # samples per spectrum (8192 ch, real-sampled)
NSPEC_PER_PKT = 2**2 # spectra per F-engine packet (fundamental GPU resolution?)
```

[2]:
```
t_pkt = 1 / f_clk * NSAMP * NSPEC_PER_PKT
N_pkt_lst_exact = t_sidereal / t_pkt
t_error = t_sidereal - int(round(N_pkt_lst_exact)) * t_pkt
print('Packets per sidereal day:  ',  N_pkt_lst_exact)
print('Fundamental drift w.r.t LST grid [s]:  ', t_error)
```

```
Packets per sidereal day:   657379840.8508301
Fundamental drift w.r.t LST grid [s]:   -1.9551996956579387e-05
```

As the calculations above show, given a sample clock of 500 MHz, we do not have an exact integral number of packets per sidereal day, leading to a $\sim 20\mu s$ drift from periodicity per day for continuous operation.

Unfortunately, we must incur even more drift. The closest integer number of packets per sidereal day (6573798841) is not even, and the next closest ($657379840 = 2^9 \cdot 5 \cdot 13 \cdot 19753$) has a large prime factor that will get in the way of achieving $t_{\text{grid}} \approx 32$ s. (We can manage to have $MCNT\_PER\_INT$ be divisible by 2^7, but only if we set $t_{\text{grid}} \approx 4$ s).

To get t_{grid} near 32 s, we need $NGRID$ in the range of 2400 to 3000, and we need to pair it with an $MCNT\_PER\_INT$ that is divisible by 2^7 (or an $NPKT\_PER\_INT$ divisible by 2^5) such that the product is near to 657379840:

$$\begin{aligned} NGRID &\in [2400, 3000], \\ MCNT\_PER\_INT \;\%\; 2^7 &= 0, \\ NGRID \cdot MCNT\_PER\_INT &\approx 657379840. \end{aligned}$$

What follows is a brute-force search for such a pairing, with t_{error} corresponding to the drift, in seconds, relative to our LST grid after a continuous day of operation.

[3]:
```
print('t_grid\tNGRID\tNPKT_PER_INT\tt_error')
for ngrid in range(2400, 3000):
    t = t_sidereal / ngrid
    npkt_per_int = int(round(N_pkt_lst_exact / ngrid))
    if npkt_per_int % 2**5 == 0:
        print('%6.2f\t%5d\t%12d\t%+6.3f' % \
              (t, ngrid, npkt_per_int,
               t_sidereal - t_pkt * ngrid * npkt_per_int))
```

```
t_grid  NGRID   NPKT_PER_INT    t_error
 35.69   2414         272320    -0.084
 35.43   2432         270304    +0.067
 34.30   2512         261696    -0.067
 34.25   2516         261280    -0.084
 34.00   2534         259424    -0.075
 33.59   2565         256288    +0.147
 33.47   2574         255392    +0.109
 33.06   2606         252256    +0.092
 33.05   2607         252160    -0.168
 32.80   2627         250240    -0.084
 32.02   2691         244288    +0.109
 31.61   2726         241152    -0.067
 31.15   2766         237664    +0.159
 30.95   2784         236128    -0.067
 30.19   2854         230336    +0.118
 29.94   2878         228416    -0.184
 29.16   2955         222464    -0.168
 28.73   2999         219200    -0.126
```

5 Results

The result above that skews the least from periodicity (${min}~ t_{error} $) while remaining close to $t_{\rm grid} \approx 32$ s is

$$NGRID = 2726,$$
$$NPKT\_PER\_INT = 241152,$$

which skews -67 ms from periodicity for each continuous day of operation. This corresponds to an integration time of $t_{\rm grid} \approx 31.61$ s.

Given that our fastest integration times are anticipated to be ~ 1 s, this is probably an acceptable amount of drift. Still, we should try to ensure that we re-synchronize the correlator each observing day.

Thus, we adopt an LST grid of $NGRID = 2726$ bins, each 31.61 s wide, indexed to 0 at $LST = LST_0 = 18h00$. Inside the correlator, this corresponds to 241152 packets per integration, with each packet containing 4 spectra, giving us a value of

$$MCNT\_PER\_INT = 964608. \quad (6)$$

What follows is example code for supporting this choice of grid.

[4]:
```python
import numpy as np

t_sidereal = 86164.0905 # s / sidereal day

NGRID = 2726
NPKT_PER_INT = 241152
NSPEC_PER_PKT = 4
MCNT_PER_INT = NPKT_PER_INT * NSPEC_PER_PKT

def calc_tspec(f_clk, NSAMP):
    '''Calculate the time for one spectrum (i.e. MCNT) in the correlator.'''
    return 1 / f_clk * NSAMP

def calc_mcnt(t, t_sync, f_clk=500e6, NSAMP=2**14):
    '''Calculate what value the correlator would have for MCNT,
    given a provided time of synchronization t_sync.'''
    t_mcnt = calc_tspec(f_clk, NSAMP)
    mcnt = np.floor((t - t_sync) / t_mcnt)
    return mcnt

def calc_t(mcnt, t_sync, f_clk=500e6, NSAMP=2**14):
    '''Calculate the time from a correlator MCNT and a sync time.'''
    t_mcnt = calc_tspec(f_clk, NSAMP)
    t = t_sync + mcnt * t_mcnt
    return t

def grid_index_from_mcnt(mcnt, mcnt_offset, start_index):
    '''Calculate the grid index from MCNT, as the correlator would.'''
    return (start_index + (mcnt - mcnt_offset) // MCNT_PER_INT) % NGRID

def calc_mcnt_offset(lst_sync, f_clk=500e6, NSAMP=2**14):
    '''Calculate MCNT_OFFSET for a synchronization pulse that
    happened at lst_sync.'''
    t_mcnt = calc_tspec(f_clk, NSAMP)
    i = grid_index_from_lst(lst_sync, f_clk=f_clk, NSAMP=NSAMP)
    lst_edge = (i + 1) * 2 * np.pi / NGRID # next grid boundary
    t_offset = (lst_edge - lst_sync) / (2 * np.pi) * t_sidereal
    mcnt_offset = int(round(t_offset / t_mcnt)) #% MCNT_PER_INT
    #mcnt_offset = int(np.floor(t_offset / t_mcnt)) #% MCNT_PER_INT
    return i+1, mcnt_offset

def lst_of_grid_index(i, f_clk=500e6, NSAMP=2**14):
    '''Calculate the true LST of the center of the LST bin
    indexed by i.'''
    lst = 2 * np.pi * (i + 0.5) / NGRID
```

```
    return lst % (2 * np.pi)

def grid_index_from_lst(lst, f_clk=500e6, NSAMP=2**14):
    '''Calculate the true grid index of the provided LST.'''
    i = np.floor(NGRID * lst / (2* np.pi))
    return i % NGRID

# Some basic tests
eps = 1e-7
for i in range(NGRID):
    lst = lst_of_grid_index(i)
    assert i == grid_index_from_lst(lst)
    assert i == grid_index_from_lst(2*np.pi/NGRID/2 + lst - eps)
    assert i == grid_index_from_lst(-2*np.pi/NGRID/2 + lst + eps)
```

[5]:
```
import time

t_sync = time.time() # arbitrary
lst_sync = 2*np.pi - 0.1 # arbitrary
# walk through MCNTs at the end of a sidereal day
start_mcnt = 2629450000
mcnt = np.arange(start_mcnt, start_mcnt+500000)
t = calc_t(mcnt, t_sync) # calculate times corresponding to these MCNTs
lst_true = lst_sync + 2 * np.pi * (t - t_sync) / t_sidereal
start_index, mcnt_offset = calc_mcnt_offset(lst_sync)
i = grid_index_from_mcnt(mcnt, mcnt_offset, start_index)
# Calculate how many MCNTs are mis-binned to cross-check the time skew.
nmiss = len(np.where(i - grid_index_from_lst(lst_true) != 0)[0])
print('Skew after 1 sidereal day: %5.4f s' % (nmiss * calc_tspec(500e6, 2**14)))
```

```
Skew after 1 sidereal day: 0.0670 s
```

5.1 Coda

If we wanted to futher suppress the periodicity error, we could consider changing the sample frequency of the correlator. Effectively, we would tweak the frequency by a small factor to ensure near-perfect peroidicity, as derived below.

[6]:
```
t_error = t_sidereal - t_pkt * NGRID * NPKT_PER_INT
print('Sample Clk [Hz]\tPeriodicity Error [s]')
print('%15.2f\t%+21.6f' % (f_clk, t_error))

f_clk_ideal = f_clk * (1 - t_error / t_sidereal)
t_spec_ideal = 1 / f_clk_ideal * NSAMP
t_pkt_ideal = t_spec_ideal * NSPEC_PER_PKT
t_error_ideal = t_sidereal - t_pkt_ideal * NGRID * NPKT_PER_INT
print('%15.2f\t%+21.6f' % (f_clk_ideal, t_error_ideal))
```

Sample Clk [Hz]	Periodicity Error [s]
500000000.00	-0.066997
500000388.78	+0.000000