A discussion on BOCF framenumber to time translation in the EVN MkIV DataProcessor

$Revision: 1.3$

Harro Verkouter

$Date: 1999/11/03 07:28:16$

Chapter 1

Introduction

In order for the Data Processor to produce scientifically valid data, it is important that the data be labelled with the correct observing time. In order to be able to select data, drive the station units and synchronize the various other parts in the system, the on-line system keeps a virtual separate clock per correlated job. The time read from such a clock is the time as it was when the experiment was being observed; the so-called reconstituted observing time. Such a clock is adressed as a ROT clock. The ROT clock measures times in units of number of system cycles (of a correlator chip) since the start of the year. Therefore, a ROT clock has a time resolution of $31.25 \cdot 10^{-9}s$, which corresponds to the duration of one system clock tick in the correlator chips which run at 32MHz.

However, this ROT clock timestamp is not tied to the data as it passes through the correlator system. In stead, together with the data, the number of Begin Of Correlator Frame (BOCF) interrupts that occurred since the last whole 10 minute interval of time, is passed. This counter is reset to zero whenever the ROT time is an integer number of 10 minute intervals. This counting takes place in the Station Unit; the SU tags the data with the current BOCF counter (in the header that is sent with the data).

At the end of the pipeline the data handler receives the data from the correlators. At this point, where the data is to be flushed to disk, the data handler faces the task of labelling the data with the correct time. This is the stage in the correlation process this document is all about.

Chapter 2

In an ideal world...

In an ideal world, the process mentioned in the introduction would proceed in the following way.

Whenever a job is to be correlated, a data handler process that is responsible for dealing with the data for this particular job is started. This data handler process receives information about the job that will be correlated in the form of the Correlation Job Descriptor (hereafter CJD). In this CJD, the **nominal** scan start time is kept. From this instance in time, the data handler can set up a BOCF number to real time translation. When data comes in from the correlators, the actual time can be computed as follows:

$$T_i = T_0 + \frac{BOCF_i}{\frac{\partial BOCF}{\partial T}}$$

where

T_i	is the time of the current integration
T_0	is the time of the last whole 10 minute time interval
$BOCF_i$	is the BOCF count found in the data for the current integration
$\frac{\partial BOCF}{\partial T}$	is the BOCF rate in BOCF/s

It is important to note that the BOCF counter increases monotonically with time and that the value of $BOCF_i$ has an upper bound, namely $(600 \cdot \frac{\partial BOCF}{\partial T}) - 1$. This is the maximum count the BOCF counter can reach during a 10 minute interval at a given BOCF rate. Every 10 minutes the

BOCF counter wraps. How the algorithm deals with this is explained later on.

It is also important to note that caution has to be taken here with respect to units. The fraction that is added to T_0 of the current interval is in units of seconds. The unit of T_0 is correlator chip clock cycles since the start of the year.

The algorithm that's needed to perform a succesful BOCF number to time translation (in a nutshell) is like this:

- Step 1: Round the nominal scan start time down to the last whole 10 minute interval of time. This is only done at initalization, when the translation has to be set up.
- Step 2: At every integration, check if the BOCF counter for the current integration has not passed its maximum value. If it has, we increment the value of T_0 by 10 minutes.
- Step 3: Calculate the time of the current integration, using the aforementioned formula.

Thus, the time values that are written by the data handler are still not of use to any data reduction package, but so is the data. Yet the times should be consistent and as exact as the system allows us (which precision is, by the way, dependant on the BOCF rate). Off-line software performs the translation of ROT time to UT time and makes sure the data is ready for shipment to the end user.

Chapter 3

Meanwhile, in the real world however...

The alert reader of the previous chapter might already have his/her suspicions aroused by the mere simplicity with which the problem was tackled.

Indeed, the real world presents us with a number of problems, assumptions, properties and behaviour in and of different parts of the correlator system. I will try to list as much as we know of/have encountered and what actions have been/should be taken in order to either circumvent or fix them.
Note: for some items it may seem at first that they bear no meaning on the subject but pray just read it. I will try to be as clear and complete as possible.

0

The data handler has no other means than the BOCF count alone as an indicator for the time of the data in an integration. There is some extra data but that is of little use as a direct measure of time.

1

One of the problems already starts in the CJD. Remember that in the CJD the **nominal** scan start time is recorded. However, data for the job may already start coming in before this time, or, data may not come in until some time after this time; you just don't know. The result of this uncertainty is that you cannot a priori initialize T_0 and forget about it. It might be that the first block of data you are receiving from the correlators contains BOCF numbers from (theoretically) one or more (in practice it is safe to assume just one) 10 minute interval(s) ahead or back with respect to the nominal scan start time: you don't know relative to which last 10 minute interval the BOCF number was counted in the SU.

The algorithm described in the previous chapter deals with this initialization problem. In the initialization stage the data handler intializes T_0 with the rounded down to the last whole 10 minute interval value of the nominal scan start time found in the CJD. Whenever the first valid datasample is received, the BOCF number found in the data is used to compute the time of the sample.

The data handler now discriminates between two cases

- $abs(T_i - T_0) < 250s$ the data handler assumes that the data indeed is coming from the same 10 minute interval it has assumed.
- $abs(T_i - T_0) \geq 250s$ the data handler needs to adjust T_0 to $\pm$ 10 minutes, depending on the sign of the actual deviation.

  a. $(T_i - T_0) \leq -250s$ apparently, the time found in the data is much less than the expected time hence it is likely that our T_0 is too low. Hence, by moving on to the next 10 minute interval should fix this.

  b. $(T_i - T_0) \geq +250s$ same reasoning as in previous case only with reversed values and comparasisons. Effectively, decreasing T_0 by 10 minutes should fix this case.

Note that if T_0 is to be adjusted it is only by $\pm$ 10 minutes. The data handler could just as easily repeat the previous procedure until the calculated time is within a range of $\pm 250s$ of the nominal scan start time. The reason why data handler does not do this is that if the deviation of the data with respect to what we expect is more than $\pm$ 10 minutes we're stuffed anyway!

2

Integrations can be missed. Due too lack of computing time, collisions on the network or any other temporarily failure.

3

The current DDD machine, jaw0, is known to hickup for up to 15-20s every once in a while during which periods the system is frozen. It is not understood yet why nor what the effects of such a hickup are on the running processes. There is some belief that some strange effects observed in the data handler (with respect to shared memory buffers) might be caused by this. This cannot be tested (I think) until we bring the real DDD machine on-line.

4

When we talk about the BOCF number of an integration that's not the whole truth. All in all, for a given sub job, the number of BOCF counters found in the data is twice the number of SU's involved in the correlator configuration in use for that particular subjob. It is the case that the SU writes in the header the BOCF count at the start of the first frame of the integration and the BOCF count at the start of the last frame of the integration. Since we have defined the time of the integration to be the midpoint we define the BOCF count for the integration as follows $BOCF_i = \frac{(BOCF_s + BOCF_e + 1)}{2}$.

Having more than one BOCF count for the integration, it is possible that not all of these have the same value (because of problems, discussed later). The data handler therefore checks for every integration if all the start BOCF counts and all the end BOCF counts found in the data are equal. If they are not, it issues a warning.

The latter case has a nasty side effect. The data handler does not know what the correct start and end BOCF counts are. Therefore it counts the occurrences of BOCF counts and assumes that the ones occurring most are the actual start and end BOCF counts.

5

Detecting a BOCF wrap. At every integration with a valid BOCF count the data handler has to check whether or not the counter did wrap. If it did we need to increment T_0 by 10 minutes. One of the most serious problems encountered in the current data handler are spurious, wrongly detected BOCF counter wraps. These clock jumps render the whole dataset for this scan useless and the only recovery possible is re-correlation of the scan.

Validation of BOCF counters. It is important to be as sure as can be about the validity of the BOCF counts. A number of checks on the BOCF counts are performed before they are declared valid.

1. BOCF counts with a value of 0 are discarded in the statistical analysis.

2. If the start and end BOCF counter value for a given interferometer are

equal (regardless of the value), this value is not taken into account.

3. After the statistical method has decided on a start and end BOCF counter value, the data handler verifies that the difference $BOCF_e - BOCF_s$ equals $t \times \frac{\partial BOCF}{\partial T}$, where t is the integration period in s.

'Keep it simple' is one of the best methods in solving complex problems. The data handler detects a BOCF counter wrap if the start BOCF counter value is less than the last known valid end BOCF counter value. This method allows for integrations to be missed and still be able to detect that the BOCF counter did wrap. If we miss more than $\frac{600}{t} - 1$ integrations this method fails (where t again is the integration period in seconds). But that means this we've lost nearly 10 minutes worth of data and so (again) we're stuffed anyway. Just for the record, the last known valid end BOCF counter value needs to initialized such that whenever the first valid integration arrives it does not falsely detect a BOCF counter wrap. This is achieved by initializing this value to 0. In this way, the condition for a BOCF wrap can never evaluate to true since 0 is the lowest possible valid value for a BOCF counter value.

6

All data for all jobs for all interferometers is contained in one block. The layout of such a block is dependant on the correlator configuration, is fixed and is **not** tied to the actual situation during correlation. Such a configuration contains (amongst others) a list of interferometers with their properties like e.g. which antennas make up the interferometer, the subband, the polarization and the location of the data for the interferometer in a data block.

The effect of this property is that if e.g. the correlator configuration is such that it is set to correlate all signals from SU 0 through to SU 7 (a full eight station correlator), every integration contains a block of data for all interferometers, even if one or both SU's delivering data for an interferometer are switched off or delivering invalid data. These last two situations of course have implications on the BOCF counter found in the data for the affected SU(s). SUs that are switched off give a zero BOCF count in the data. Under certain circumstances (and this has happened) the number of zeroes as BOCF count outnumbered the actual valid BOCF and hence the statistical method decided that a BOCF count of zero was most propable and therefore ruining the BOCF to time translation. The statistic gathering method now discards all BOCF counts with a value of zero. If, eventually, the actual BOCF count is zero, the integration will be skipped because of non-valid BOCF numbers. An integration will be flushed to disk if and only if two valid BOCF counts (start and end) were found in the data.

7

Each data block received from the correlators also contains the value of the SYSTICK counter at the time of integration. The SYSTICK is a systemwide one tick per second counter. This information is used by data handler to attempt to detect if it is missing one or more integrations. For every valid integration that data handler processes it calculates an expected value of the SYSTICK, based on its last recorded value and the current integration period. Of course it skips the first valid integration, in this case it just saves the value of the SYSTICK.

If the expected value of the SYSTICK is not equal to the one found in the data, there's two possibilities,

- either the SYSTICK found in the data is larger than the one we expected; this means we have missed one or more integrations,
- or, the SYSTICK found in the data is less than the one we expected; in this case we are in trouble since it means that we're reading an old block of data.

In both of these cases the data handler prints out an appropriate message and syncs up. It means that the internal last recorded value is set to the value in the data. Apparently, I have discovered a bug here! The logic should read of course

- if the SYSTICK value found in the data is larger than expected, update our internal value with the one found in the data
- otherwise update our internal value with the expected value!

This explains why, if the data handler detects a 'repeat systick', it never recovers!

In the current version of data handler nothing else is done with the SYSTICK value. Theoretically, this value might be used as a means of predicting the next BOCF count that should be found. Care has to be taken here since integrations may be missed. Also, the BOCF counter wrap should be taken into account. These extra checks may work fine but what happens if we go to integration periods $< 1s$? The SYSTICK has only a time resolution of $1s$.

The use of the SYSTICK value might help cure the current problem with erroneously detected BOCF counter wraps since it gives yet another check on the validity of the BOCF counter values. However, I'm not fully convinced that this is the core of the problem.

8

Individual interferometers are not flagged as valid or invalid. This would help in the statistical analysis of the BOCF counter values. If data handler knows that one or more SUs are either off-line or producing garbage it could discard the BOCF counter values for interferometers where data is combined from one or more of the invalid SUs thus hopefully yielding a better statistic.

9

It was brought to my (Harro's) attention that apparently, the data found in the correlation frame header actually apply to the next frame in the data. This means that currently data handler is mis-interpreting the framenumbers found in the data. The effect of this is that all the times calculated by data handler are off by 1 frame (how much that is in time is dependant on the BOCF rate).

10

It might be necessary to have a closer look at the cdi (CorrelatorDataInterface) and/or the shared memory classes in the on-line system. As yet we do not have an explanation for the occasional occurrence of a 'repeat systick' event in the data handler. This means that the data handler is reading a block of data from the past. Given the methods used by data handler to get at the data and the way the cdi is supposed to deliver data this should not happen.

Chapter 4

The future

After some discussion it was quite clear that the preferred method of time-stamping the data should be such that a unique timestamp be stuck to the data and flows through the system together with the data. This will enable us to look at the data only and say unambiguously: the time for this integration is such-and-such.

It is unpropable that we will arrive at this situation. There are just not enough bits in the correlator frame header to accomodate this.

A number of suggestions came up during the discussion;

0

In the correlator frame header, 5 or 6 bits are reserved for future use. We can use this bits to let the SU fill in how many times the BOCF counter did wrap since the nominal scan start time. This would mean that data handler does not have to detect the wraps itself; the information is carried in the data. It still does not solve the problem of detecting if the framenumbers (or header in this case) are(is) coming through correctly.

For this to work, it involves work by Phil, Albert and Harro. Phil needs to count the wraps and write it in the header (this was already discussed with him and according to Phil it would not be a long task; on the order of hours). Albert would have to make changes such that the headers come through the system (or at least include this number of wraps in the output header). I would have to update data handler to deal with this.

All in all quite a big operation and certainly not expected to be executed before the end of this year. It just might be the best long term solution that we can have, given the fixedness of the correlation frame header.

Another big advantage of this solution is that it also gives us more and better diagnostics. The possibility of getting the full correlator frame headers enables far better detection of where along the pipe-line things may have gone wrong. It can easily be detected if a SU is failing or if e.g. a correlator chip is garbling the header.

1

Use external messages which originate from a SU that link current ROT and BOCF counter. This could be used as a diagnostic to check if the data handler is still in sync with the ROT clock. If it is not the operator can be alerted so the job can be cancelled. At the moment erroneously detected BOCF counter wraps are usually detected off-line which is, needless to say, even more inefficient than just having to re-correlate the job. This can be done on a relatively short timescale.

2

Add some more checks/maybe more sophisticated checks to the statistics routine in order to minimise the propability of falsely deciding on the current BOCF count. This can be done on quite a short timescale.

3

Check the code in the on-line system regarding the 'repeat systick' clause. Maybe needs patching of the release but may solve part of the current problems. Very short time scale.

4

It might be useful to find out how the designers of the system had envisioned to do the time stamping of the data. These are the people that decided on the information content of the correlation frame header. It is unlikely that they did not have something in mind about how to find out the correct timelabels at the end of the data pipeline.