NATIONAL RADIO ASTRONOMY OBSERVATORY
Charlottesville, Virginia

13 November 1984

To: A. R. Whitney
From: L. R. D'Addario
Subject: Comments on "Tape Synchronization", Acq Memo #27

1. SEPARATION OF FUNCTIONS

The above referenced memo gives few details of the interaction of the correlator control computer (CCC) with the DPS, but from the scenario presented it seems that an important principle may be being overlooked. The commands from the CCC should be clearly separated into two kinds: control of the signal interface and control of the tape transports. In my notes of 21 Feb 84 (recently distributed as Acq Memo #24), I suggested physically separate interfaces for these functions; in subsequent discussions, it was agreed that the data rates involved did not justify two physical interfaces, but that the functions could be kept logically separate while sharing a single line. I am concerned that this logical separation may get lost. The main point to keep in mind throughout the design is that we may not always be using tape. As much as possible should be made independent of the data transmission technology, and this part should be kept rigorously separate from the technology-dependent functions.

Examples of signal interface functions include: (1) select digitizer channel to be connected to a given interface channel; (2) set data rate for each channel (which may differ from the original sampling rate); (3) set time of a specified sample on each channel, and parameters of update formula for future samples. For functions of this type, it should be completely transparent to the CCC that the data are coming from tape. No reference to tape transports should be involved.

Examples of transport control functions include: (1) ensure that the proper tape is mounted on a given transport, and that it is positioned at a "footage" which is reasonable; (2) monitor the operation of the transport, including tape position, speed, BOT/EOT, etc.; (3) allow overriding of transport controls normally handled automatically by the DPS, for test purposes or for intervention in case of failures.

An interesting question is how to handle a tape change; assuming that the new tape is on a different transport, we would like to accomplish a nearly-instantaneous switchover. It would be logical to have the switch occur in the DPS, so that data from the same station continues to be provided on the same interface channels. But if Marty Ewing's "global bus" is implemented in the correlator, then it becomes easy to do the switching there, and it seems wasteful to provide the switching in two places. Can the fact that different tape transports are involved still be made transparent to the signal interface controls? Yes; the CCC need only know that at a certain pre-determined time the channels of a given station will move from one DPS to another. For some time before the changeover, the CCC can send duplicate timing commands to both DPSs; then, at the right time, the switches in the correlator are thrown. The validity signal will come true on the new DPS and go false on the old DPS, perhaps with a gap (neither valid) or with an overlap (both valid). Notice that the CCC is not concerned about the cause of the changeover (tape-related); maybe the signals are coming through satellites and we need to switch from Satellite A to Satellite B because A is scheduled for maintenance today. (Meanwhile, of course, the transport control section of the CCC has been setting things up so that the changeover will be possible. But these functions are well separated; the transport control stuff could be replaced by, say, the satellite receiver control stuff.)

2. "WALL CLOCK" TIME

It is worth pointing out that requiring all DPSs as well as the correlator to keep yet another kind of time is an inessential complication. It may lead to some convenience, so perhaps it should be done, but it needs more careful consideration. First of all, it does not make the communication between the CCC and DPS completely asynchronous. As an extreme example, the CCC could not pass all the timing commands for the next three days in one big burst; the DPS will have a finite command buffer, for one thing. Besides, keeping too far ahead will require maintaining fancy facilities that we really don't want: we would have to be able to get into the DPS buffer to edit previously-sent commands, in case we change our minds. Also, the CCC needs to operated in real time, in that it must keep up with the correlator hardware; so it must compute DPS timing commands at a constant average rate. So, for various reasons, commands which reference the "wall clock" will still have to be sent on a regular, synchronous schedule.

The examples in AM 27 seem to indicate that new timing commands are contemplated every 10 sec, and that these would be sent separately, one during each 10 sec interval. They could, of course, be sent in bursts, say 10 at a time covering the next 100 sec, provided that the DPS has enough buffering. In any case, the CCC and the DPS must come to an agreement as to how often and in what size blocks the timing commands are to be updated. Having made such an agreement, the need for "wall clocks" disappears. Instead, one distributes throughout the room a periodic timing signal whose period is this agreed update interval, and all commands are effective on the next rising edge of this signal. The period could even be changed in the future if necessary, though we would be wise to choose a good value now. In the (unlikely) event that it becomes necessary to send several commands that must take effect at different times during an update period, then each could be tagged with the number of 16 MHz clock periods beyond the next timing edge when it becomes effective; if you like, this could be called a "wall clock" whose length is one update period (say 1 to 100 sec).

Unless it can be shown that the CCC cannot possibly maintain an update schedule with a period somewhere between 1 and 100 sec, the complexity of the time-tagged command scheme does not seem justified. Major simplifications of the DPS software will result if each command is executed on the next timing edge after its receipt. We have enough problems in the VLBA making various computers talk to each other without making this interface unduely complicated. A rigorous, periodic protocol is much simpler than the one proposed in AM 27.

3. FORMAT OF TIMING COMMANDS

The scenario of AM 27 suggests that the time of a sample would be the sum of a reference time and a "delay offset", sent separately, with the latter in the form of "polynomial coefficients." It is important that the algorithm recognize the inherently integer nature of the timing; the time that a sample is provided can only be changed from the sampling time by an integral number of sample periods. There should be no way in the world that roundoff errors could affect the result, regardless of the arithmetic precision in the DPS or CCC. I assume that the polynomial mentioned has time as its independent variable; in that case it will certainly be subject to errors which depend on the precision of the coefficients and of the arithmetic. A much better approach is to send (a) the exact time of a certain sample; (b) the number of samples thereafter when a delay of +1 or -1 sample is to be inserted; (c) the periodic number of samples P after which additional delays of +1 or -1 are to be added; (d) [if necessary] the number of length-P periods after which the value of P is to be incremented or decremented; etc. Such a scheme is rather straightforward to implement in hardware.

4. USE OF VALIDITY FLAG

The suggestion that each timing command should carry an expiration time is interesting, but probably not necessary. In any event, reaching the expiration time should not be signaled by using the data validity flags. The latter should indicate merely that the DPS is successful in delivering the requested data, and this should not be confused with whether a reasonable request has been received. The latter has to do with communication on the control interface, which should have its own validity checks; failure of the DPS to receive commands when expected should cause a warning or error message to be sent to the CCC on the command interface, but the data interface should continue providing valid data based on the last command received.