REAL-TIME PROCESSING OF DOWNLINK PHASE MEASUREMENTS

LARRY R. D'ADDARIO

National Radio Astronomy Observatory*
2015 Ivy Road, Charlottesville, Virginia 22903, U.S.A.

March 11, 1992

INTRODUCTION

The two-way time transfer link between earth station and satellite will be of the form shown in Figure 1. The downlink signal will be converted to a convenient I.F. and then its phase will be measured with respect to a programmable reference. Some details of the analog signal processing leading up to the phase detector were discussed in OVLBI-ES Memo 22. In our implementation, the phase detector is an analog device consisting of two mixers operating in phase quadrature. Outputs of the two mixers will be sampled and digitized at a rapid rate and the resulting data will be processed in real time by a digital signal processor. The processed data will be presented at a much slower rate to the main station control computer, which could perform additional real-time processing if necessary, but which is responsible mainly for writing the data to a log file. Off-line (non-real-time) software will convert the log file to the forms needed by correlators and orbit determination centers.

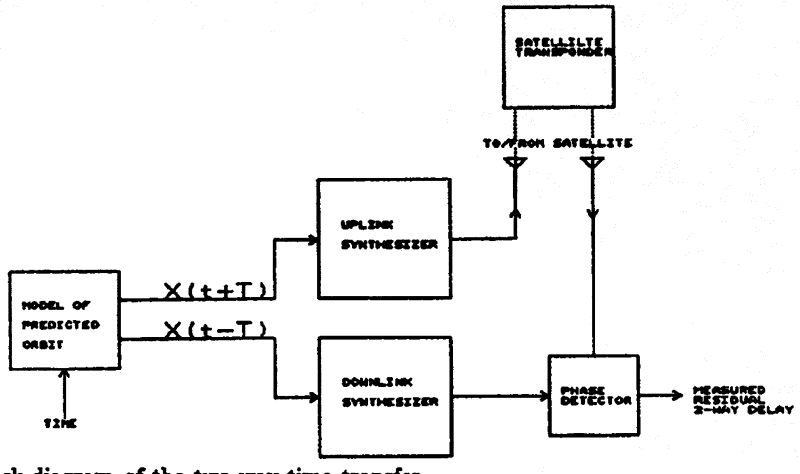


Fig. 1: Conceptual block diagram of the two-way time transfer.

The present memo is concerned with the design of the required real-time digital signal processing.

SAMPLING RATES

As mentioned in OVLBI-ES Memo 22, the highest feasible input sampling rate will be supported subject to a minimum requirement of 1 kHz. The calculations below will consider sampling rate of 10 kHz. At the DSP output (to the station computer), we assume a rate of 10 measurements per second. This is expected to be adequate for representing the fastest fluctuations in the signal phase, which are expected to be due to fluctuations in the tropospheric delay.

---

*The National Radio Astronomy Observatory (NRAO) is operated by Associated Universities, Inc., under a cooperative agreement with the National Science Foundation.

Processing of Individual Samples

Certain calculations must be done for each individual sample:

1. Accumulate data on phase detector errors
2. Correct for phase detector errors
3. Convert measurement from rectangular (x, y) to polar (a, ϕ) form
4. Resolve cycle ambiguity (i.e., determine whether a cycle has been completed since the preceding sample, and if so increment or decrement the integer part of the phase)

The result of these calculations will be a single number equal to the measured phase. It needs to have a resolution of better than 0.1 radian (.016 cycle, 6 bits after binary point) to avoid significant phase noise being added, and a range sufficient to cover the maximum variation in orbital position error during a tracking pass. The latter is predicted to be $< \pm 1$ km, but larger errors might occur in the early phases of the missions. At a wavelength of 2 cm (VSOP), ± 1 km gives a two-way phase error of $\pm 10^5$ cycles, requiring 18 bits for the integer part of the phase. Thus, a 24-bit fixed point representation would be adequate but additional bits would be desirable.

The analog phase detector operation may be modeled as

$$\begin{aligned} x &= g_x a \cos(\phi) + x_0 \\ y &= g_y a \sin(\phi + \epsilon) + y_0 \end{aligned} \tag{1}$$

where x, y are the phase detector outputs; x_0, y_0 are d.c. offset errors, g_x, g_y are scale factors; and ϵ is the error in phase quadrature between the x and y measurements. The signal being measured has amplitude a and phase ϕ. Equations (1) are easily solved for a and ϕ:

$$\begin{aligned} \hat{x} &= (x - x_0)/g_x \\ \hat{y} &= [(y - y_0)/g_y - \hat{x} \sin \epsilon]/\cos \epsilon \\ a &= \sqrt{\hat{x}^2 + \hat{y}^2} \\ \phi &= \tan^{-1}(\hat{y}, \hat{x}) \end{aligned} \tag{2}$$

where the arctangent function is the four-quadrant version (2π radian range, ATAN2 in Fortran). The parameters x_0, y_0, g_x, g_y, and ϵ can be determined by separate calibration measurements using a known test signal, but they can also be derived from a series of measurements on the unknown signal provided that they are relatively constant and that the signal amplitude a is also relatively constant. The process is simplified if $\epsilon = 0$ to sufficient accuracy. In our implementation, the quadrature reference signals to the phase detector have separately programmable phase shifts which should allow us to make $\epsilon = 0$; tests will determine whether this is accurately achieved over a range of signal frequencies and over temperature and time. To allow for determining the calibration parameters, the DSP will accumulate the sum of x, y, x^2, y^2, and xy over a programmable interval. These will be available to the station computer.

Resolution of the cycle ambiguity can be accomplished as follows. For each sample, let $-\pi < \phi \leq \pi$, and let the previous sample value be ϕ_{old}. Then calculate $\delta\phi = \phi - \phi_{\text{old}}$; if $\delta\phi > \pi$ then decrement the integer part of the phase; if $\delta\phi < -\pi$ then increment the integer part of the phase; otherwise leave the integer part unchanged. Note that in the presence of noise this algorithm can sometimes make the wrong decision. In that case, an unrecoverable cycle slip occurs. The probability of such a cycle slip must be kept extremely low, and the

required signal-to-noise ratio depends on the ratio of the signal frequency to the sampling rate; the appropriate theory is developed in Appendix A. We proceed by assuming that the SNR is adequate.

The above algorithm becomes particularly simple when ϕ is represented in units of cycles using fixed point, twos-complement binary with the binary point just after the sign bit. Then the integer part is changed if and only if the $\delta\phi$ subtraction causes an overflow; and the decision to increment or decrement depends only on the sign of $\delta\phi$.

Filtering

The above processing may result in 1000 to 10 000 phase measurements per second, each being a 24-bit (or larger) number. Once the phase ambiguity has been resolved, it is believed that the intrinsic bandwidth in this time series is no more than a few Hz. This is true in spite of the fact that over an interval of a 1 sec the measured phase may change by hundreds of cycles; this large time derivative can result from a component (viewed in the frequency domain) whose amplitude is around 10^5 cycles but whose frequency is the order of 10^{-4} Hz. Such components are expected to result from position errors that vary on time scales related to the orbital period. If the signal is passed through a linear lowpass filter whose bandwidth is several Hz, components at such low frequencies will be negligibly affected, despite their large amplitudes, provided only that numerical errors (roundoffs and overflows) are properly controlled.

We therefore plan to process the individual high-rate samples, after cycle resolution, by passing them through a digital, finite-impulse-response lowpass filter of bandwidth 5 Hz and then re-sampling at 10 Hz. An FIR filter with symmetrical coefficients is always a constant-delay filter, which is a desirable feature in our application. Filtering and resampling are efficiently implemented together, forming what is known as a decimation filter.

Suppose that the filter is required to have less than 1.0% ripple in a passband of 0 to 4.0 Hz, and at least 40 dB of rejection in a stopband beginning at 5.0 Hz. Then to decimate from a rate of 1 kHz to 10 Hz (decimation factor $M = 100$) requires a filter length of about $N = 1800$ coefficients. With similar specifications, decimation from 10 kHz to 10 Hz ($M = 1000$) would require $N \approx 18000$. Such long filters are impractical to implement; they require too much memory and computing time, and would accumulate excessive numerical errors. However, there is a way to implement the filter in several stages so as to avoid these problems [1]. Consider the following three filters in cascade. For each one, the passband extends from 0 to 4.5 Hz with ripple less than 0.5%, and the stopband rejection is 40 dB.

I. Decimate 10 kHz to 500 Hz, stopband starts at 495 Hz; needs $N_1 = 49$ coefficients.
II. Decimate 500 Hz to 50 Hz, stopband starts at 45 Hz; $N_2 = 39$.
III. Decimate 50 Hz to 10 Hz, stopband starts at 5 Hz; $N_3 = 115$.

The output of each stage is the input of the next. The overall result is a decimation from 10 kHz to 10 Hz with the desired response, and a total of $N_1 + N_2 + N_3 = 203$ coefficients is needed (vs. 18000 with a single stage). We will use this approach in our implementation.

Signal Processors

We have so far been experimenting with the Analog Devices ADSP-2100 series of digital signal processor chips; in particular, we are using the ADSP-2105 in our laboratory breadboard. The overall architecture of this processor is shown in Figure 2. This did not result from any rigorous selection process, but rather from the fact that a single-board computer using this processor was available at low cost along with some development software. The device appears to be adequate for our purposes, although the choice is not final.

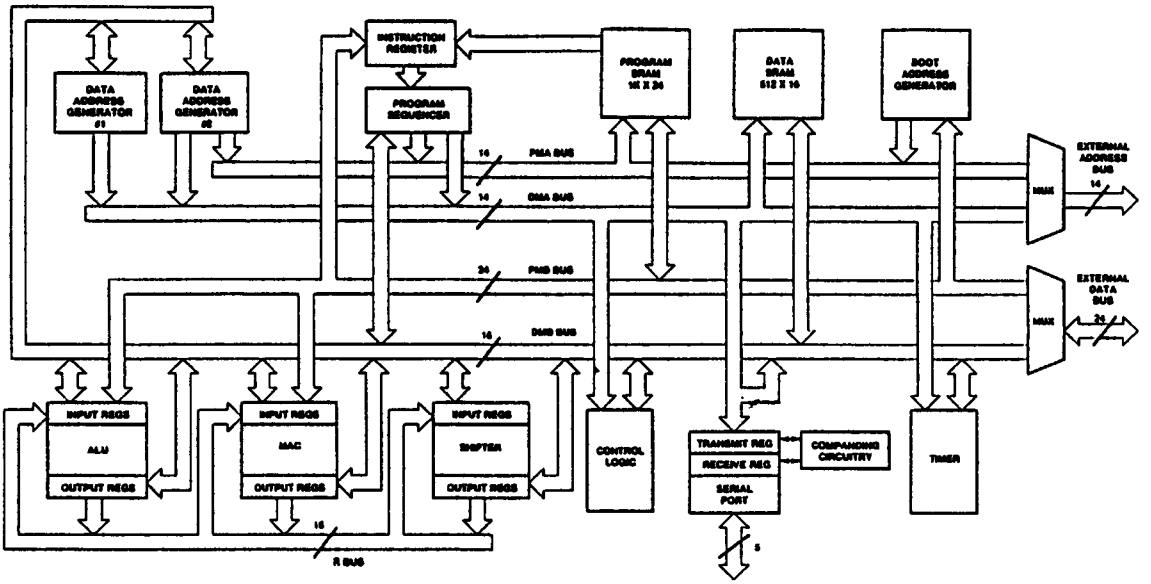


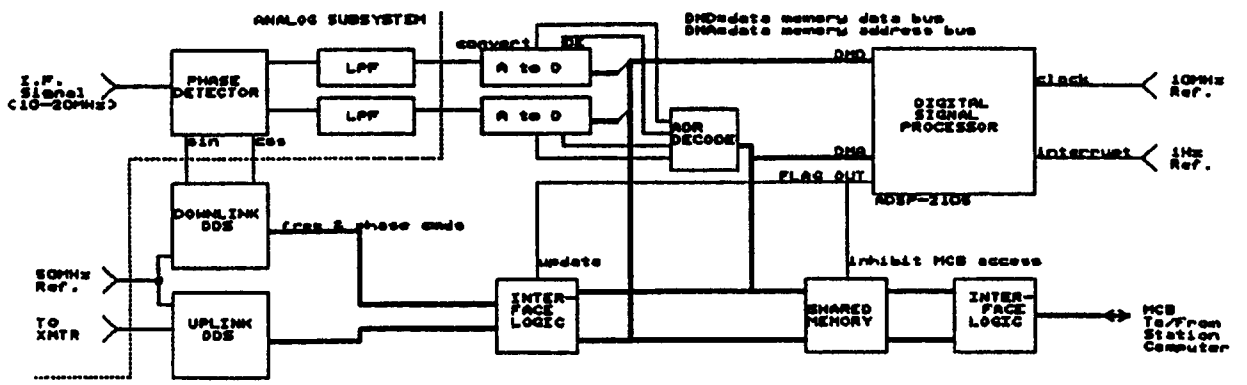
Fig. 2: Block diagram of the ADSP-2105 processor (from [2]).

The ADSP-2105 is a fixed-point processor with 16-bit data words and 24-bit program words in separate memories. All instructions execute in exactly 100 nsec (1 cycle of the external clock), and calculations can be pipelined so that several things happen in the same cycle. For further details, consult the manufacturer's literature [2,3]. Calculations of computing times given below are based on the use of this processor.

Implementation Details

Hardware

A block diagram of the prototype hardware configuration is shown in Figure 3. Timing of the entire subsystem is controlled by the digital signal processor; its 10 MHz clock is taken from the station timing reference (H maser), and it receives 1 Hz interrupts synchronized to UTC seconds. The DSP contains a programmable timer that enables any frequency between these to be synthesized. In particular, the sampling frequency is generated and used to trigger conversions in two analog to digital converters connected to the (filtered) phase detector outputs.



Two-Way Timing System:
DIGITAL SUBSYSTEM BLOCK DIAGRAM

Fig. 3: Digital subsystem block diagram.

The DSP also controls the two direct digital synthesizers. Present plans are to update both of them once per downlink phase sample; although this is more often than necessary, the CPU cycle requirements are modest. Further discussion of the DDS programming will be given in a separate report.

Communication with the station computer is via the VLBA Monitor/Control Bus (like all of the station hardware). A shared memory of 16-bit words is addressable by both the MCB and the DSP, and is used to pass control parameters to the DSP (mainly the DDS frequency parameters) and to pass the downlink phase measurement results to the station computer. Diagnostic information and calibration results are also passed. The DSP operates most of the time with instructions and data stored in on-chip memory, so that access to the shared memory by the MCB does not affect execution of DSP instructions. But if a DSP instruction requires access to external memory, including the shared memory and the ADCs (which are memory mapped), it must wait during any MCB access.

To ensure precise timing of critical events like the phase detector sampling and the DDS updating, MCB access can be inhibited under software control. With this precaution, and considering that all DSP instructions execute in exactly one cycle and that the code is deterministic, the sampling times and DDS update times are predictable with a resolution of one DSP clock cycle (100 nsec) and a precision limited only by the clock jitter (probably a few nsec).

The DSP excecutable code is normally stored in an EPROM and is automatically loaded into the on-chip program memory during the power-up sequence. Under software control, the processor can be re-loaded with different code from the EPROM. Our configuration also makes it possible to download code from the station computer over the MCB.

Firmware

An outline of the required code is given in Table 1. Calculations and simulations show that there should be little difficulty in completing the required computations in the available time, even for the 10 kHz sampling rate. Here are some of the timing estimates:

Operation	CPU Cycles	Samples/Oper.
Trigger next sample; read ADCs	6	1
Accumulate calibration data	28	1
Correct for phase detector errors	11	1
Arctangent (maximum)	95	1
Resolve cycle wrap (maximum)	10	1
Update uplink and downlink synthesizers	67	1
Misc. flow control and test instructions	15	1
Filter stage 1		
Copy data to buffer	$4M_1 + 7$	M_1
Compute output	$3N_1 + 19$	M_1
Filter stage 2—compute output	$3N_2 + 19$	M_1M_2
Filter stage 3—compute output	$3N_3 + 19$	$M_1M_2M_3$

The M_i are the decimation factors and the N_i are the lengths of the filter stages. The filter computation cycle counts are based on double-precision calculations (32-bit numbers) using code supplied by Analog Devices [3]. The other times are based on counting instructions in actual code (preliminary, untested version). These numbers are believed to be quite accurate.

As discussed earlier, we choose $M_1 = 20$, $M_2 = 10$, and $M_3 = 5$ to achieve a total decimation of 1000 from 10 kHz to 10 Hz. Because of the memory addressing facilities built into the ADSP-2105, it is efficient if the filter lengths are of the form $2^n - 1$; so we choose

Table 1: Outline of DSP Firmware

```
Initialize (on powerup or reboot):
        Set up timer interrupt at default sample rate
        Set up external interrupt (from 1 Hz tick)
        Initialize all registers
        Set GO_MODE, other internal modes
        Initialize FIR filter
        Drop into main loop

Main Loop:
        /* All of the code here is interruptable */
        IDLE  /*wait for interrupt*/
        if (1-Hz-flag is set)                  /* do this once per second */
           check FIR filter sync; reset if necessary.
           copy parameters from external shared RAM.
           reset timer period for specified sampling rate.
           re-load system control register. /*may force re-boot*/
           reset 1-Hz-flag.
        /* The 3-stage FIR filter decimates the sampling rate by a
           factor of M1*M2*M3. */
        if (M1 samples taken)                  /* do this every M1 samples */
           copy M1 samples from input buffer to filter stage 1 input.
           compute one result from filter stage 1.
           if (M2 results from filter stage 1)./* every M1*M2 samples */
              compute one result from filter stage 2
              if (M3 results from filter stage 2)./* every M1*M2*M3 s.*/
                 compute one result from filter stage 3.
                 write result to shared memory.
        goto Main Loop.

External interrupt (1 Hz):
        /*Unfortunately this interrupt has highest priority, so
         we must minimize time spent in it.  Code in-line at vector.*/
        Reset timer to zero. /* synchronize timer to ext 1Hz */
        Set internal 1-Hz-flag.
        Return from interrupt.

Timer interrupt:
      Inhibit bus requests from MCB.     /*reset FlagOut pin*/
      Send sync signal to DDSs.          /*reset FlagOut pin*/
           /*this set them to the previously-stored frequencies*/
      Start next A to D conversions for x,y.
      Read x,y from ADCs.
      Allow bus requests from MCB.       /*set FlagOut pin*/
        /*even if MCB is in middle of a request, the inhibiting is so
        brief that there should be no loss of data; DEVREQ lasts 40us.*/
      Accumulate  sum(x), sum(y), sum(x^2), sum(y^2), max(x), min(x),
                  max(y), min(y).
      Correct x,y for offset and gain errors.
      Compute phi=atan2(y,x).
      Resolve cycle wrap; increment/decrement integer part P.
      Convert phase result P.phi to s15.16 format.
      Store phase result in circular input buffer of length M1.
      Calc uplink synthesizer freq for next interrupt; send to synthesizer.
      Calc dnlink synthesizer freq for next interrupt; send to synthesizer.
      Decrement integrator-time.
      if (integrator-time == 0)
         compute  signal amplitude  sqrt(<x^2+y^2>),
                  x,y  peaks,
                  x,y  offset.
         set status bits.
         reset integrator-time.
      Return from interrupt.
```

$N_1 = 63$, $N_2 = 31$, and $N_3 = 127$ rather than 49, 39, and 115, respectively, which would just meet the specifications.

At the 10 kHz rate, there are 1000 CPU cycles available per sample (100 nsec/cycle). Using the values of M_i and N_i just selected, we find that at most 232 cycles are needed for the code that executes every sample, plus 255/20 cycles per sample for the first stage filtering, and 112/200 and 112/1000 cycles per sample for the respective other stages. Thus a total of 246 cycles per sample will be used, or less than 25% of the available cycles.

Present Status

All of the digital hardware shown in Figure 3 exists in breadboard form and has been tested successfully. Some experimental code for the DSP has been written and tested, including nearly all of the processing that must be done for each sample. The three-stage FIR filter has not yet been coded; a single-stage version has been written but not yet tested.

REFERENCES

[1] R. E. Crochiere and L. R. Rabiner, *Multirate Digital Signal Processing.* Prentice-Hall, 1983.

[2] Analog Devices Corp., "DSP microcomputer: ADSP-2105." Data sheet, October 1990 (P.O. Box 9106, Norwood, MA 02062; 617/329-4700).

[3] Analog Devices Corp., A. Mar, ed., *Digital Signal Processing Applications Using the ADSP-2100 Family.* Prentice-Hall, 1990.

Appendix A: Required Signal-to-Noise Ratio

Let $\phi = (\phi_t + e) \mod 2\pi$ be the result of the most recent phase measurement, where ϕ_t is the true value of the phase and e is the measurement error (due to system noise). Similarly, let $\phi' = (\phi'_t + e') \mod 2\pi$ be the result of the previous phase measurement. Then the change in measured phase is $\delta\phi = \phi - \phi'$ and this quantity will be used to determine whether a cycle has been completed according to the algorithm

$$\begin{aligned} &\text{if} \quad \delta\phi > \pi \quad \text{then} \quad i := i - 1; \\ &\text{if} \quad \delta\phi < -\pi \quad \text{then} \quad i := i + 1 \end{aligned} \tag{A1}$$

where i is the cycle count. The total measured phase is then

$$\theta = 2\pi i + \phi.$$

We say that a *cycle slip* occurs if θ differs by more than π from the value it would have had if there were no noise ($e = e' = 0$). The algorithm is based on the assumption that the true value of θ will never change by more than π from one sample to the next; this will be the case for sampling at or above the Nyquist rate in the absence of noise. A cycle slip occurs when the added noise causes the measured data to violate this rule. Thus, a cycle slip occurs whenever

$$|(\theta - \theta') + (e - e')| > \pi \tag{A2}$$

which implies

$$|\theta - \theta'| + |e - e'| > \pi \tag{A3}$$

and

$$|e - e'| > \pi - |\theta - \theta'| \tag{A4}$$

To limit the expected rate of cycle slips to $1/T$, we require

$$\Pr\{|E| > \pi - |\delta\theta|\} < 1/(f_s T) \tag{A5}$$

where $E = e - e'$, $\delta\theta = \theta - \theta'$, and f_s is the sampling rate. This assumes that all errors are independent and identically distributed and that $\delta\theta$ is nearly constant (i.e., the signal is quasi-sinusoidal). (For example, at $f_s = 1\,\text{kHz}$ and $T = 1\,\text{hour}$, the probability must be $< 2.78 \times 10^{-7}$.)

If the signal whose phase is being measured is

$$a\sin(2\pi ft) + n(t)$$

where $n(t)$ is a zero-mean, stationary random process with $\langle n^2 \rangle = \sigma^2$, then any one measurement of the phase will have an error whose distribution is given approximately (for small errors) by

$$\Pr\{|e| > \epsilon\} = 0.5\,[2 - \operatorname{erfc}(a/\sigma)]\operatorname{erfc}((a/\sigma)\tan\epsilon) \tag{A6}$$

where $\operatorname{erfc}(\cdot)$ is the complementary error function. (The proof of this is omitted here for brevity, but can be supplied by the author upon request.) If the noise on successive samples is independent, then we can substitute E for e on the LHS of (A6) by substituting $\sqrt{2}\epsilon$ for ϵ on the RHS. Since $\delta\theta = 2\pi f/f_s$, we have $\epsilon = \pi(1 - 2f/f_s)$. With these substitutions, and assuming that $a/\sigma > 2$, (A6) becomes

$$\Pr\{\text{cycle slip}\} \approx \operatorname{erfc}[(a/\sqrt{2}\sigma)\tan\pi(1 - 2f/f_s)] \tag{A7}$$

and this can be evaluated for cases of interest. A few results follow.

$$\begin{aligned}
(a/\sigma)^2 = 10, &\quad f/f_s = 0.25 \Rightarrow p = 1.34 \times 10^{-23}. \\
(a/\sigma)^2 = 10, &\quad f/f_s = 0.4 \Rightarrow p = 0.22 \\
(a/\sigma)^2 = 30, &\quad f/f_s = 0.4 \Rightarrow p = 6.5 \times 10^{-5} \\
(a/\sigma)^2 = 100, &\quad f/f_s = 0.4 \Rightarrow p = 3.3 \times 10^{-13}
\end{aligned} \tag{A8}$$

From this it can be seen that for signals whose frequency is less than half the Nyquist rate ($f/f_s = 0.25$), low SNRs ($< 10\,\text{dB}$) are adequate; but as the Nyquist rate is approached things get rapidly worse, so that at 80% of Nyquist ($f/f_s = 0.4$) the SNR needs to be nearly 20 dB to keep the rate of cycle slipping to less than one per hour.