Theory, Operation, and Computer Programming of the 512-Lag Correlator System

R. F. Jurgens
Communications Systems Research Section

The theory, operation, and computer programming of the JPL–Haystack autocrosscorrelator system is described. This system provides 512 channels with a basic bandwidth of 10 MHz and has provisions for multilevel sampling, oversampling, and bandwidth multiplication by factors of 2 and 4. The system controller features automatic word extension to 40 bits, precise control of the zero lag counter, and computer-enabled run or start commands.

I. Theory and Operation of JPL–HAC Correlator System

A. General Description of the Autocorrelator and Controller

The JPL–HAC 512 channel correlator is a high-speed digital signal processor designed to measure auto- and crosscorrelation functions. The discrete Fourier transforms of these functions are power spectra and cross power spectra, respectively, when certain corrections for the digital quantizing of the signals have been made. In particular, the correlation function $R_{xy}(\tau_n)$ may be formed from sample functions of the signals $x(t)$ and $y(t)$, where x and y are real or complex as shown below:

$$R_{xy}(\tau_k) = \sum_{n=1}^{N} x(\tau_n)y^*(\tau_{n+k})$$

The samples of $x(t)$ and $y(t)$ are usually assumed to be zero mean, gaussian random variables generated from two ergodic processes. In this case, the values of $y(\tau_{n+k})$ are assumed to exist for all values of $n + k$ for which there is interest. For digital processing, x and y are quantized variables. The most crude approximation results when only two levels are preserved, i.e., only the sign of the variable is measured and is represented by binary levels 0 and 1. Other quantizing systems are possible, however. Measurements of power spectra obtained from the two-level processing (one-bit correlations) are degraded in signal-to-noise ratio by only a factor of $\pi/2$ relative to the signal-to-noise ratio obtained from continuous variables. This mode is particularly easy to implement since the product can be formed by simple correlation matching using an exclusive OR gate. Specifically, the quantized signals x_n and x_{n+k} are the present and delayed samples of two real processes $x(t_n)$ and $y(t_{n+k})$,

$$\bar{x}_n \oplus y_{n+k}$$

where $\bar{x}_n$ is the logically inverted signal and $\oplus$ indicates the exclusive OR operation. A logic 1 state from this gate represents a correlation match and is counted for each n in the sequence.

The JPL–HAC 512 channel autocorrelator is an implementation of one half of the 1024 channel autocorrelator designed by Jim Levine for use at the Haystack Observatory. The JPL autocorrelator system has been modified slightly to provide a precisely measured run time that is programmable by the control computer.

The autocorrelator system consists of the 512-channel autocorrelator (8 subsystem units with one spare unit), a control unit and computer interface module, a data sampler and signal drive unit, a PDP-11/20 control computer, and a direct memory access (DMA) link to the SDS-930. A block diagram of the system configuration is shown in Fig. 1.

Special features of the system presently include a computer controlled run or start, 16-bit word extension by automatic DMA from the controller permitting integration times as long as 10 hours at the maximum clock rate, program controlled bypassing of defective autocorrelator units, and data unloading and restarting in less than 30 ms for 512 channels.

B. Autocorrelator Operation Modes

The sensitivity of the usual 1-bit (2 × 2 level) autocorrelator can be improved by sampling faster than the Nyquist rate or by extending the number of quantizing levels of the signal (multilevel correlation). Table 1 shows some degradation factors relative to a continuous system (no quantizing) for various possible systems.

The table indicates that not much is gained beyond double sampling or 3 × 3 levels. The JPL–HAC autocorrelator ultimately may be operated in the modes indicated by Table 2.

The unmodified autocorrelator cards also permit 3 × 5 level processing in some modes. However, this facility has been eliminated temporarily in order to provide an extended run gate which permits the autocorrelator to properly complete the overflow scanning sequence.

The simplest mode of operation is the (2 × 2) level mode with sampling at the Nyquist rate. Figure 2 shows diagram of the autocorrelator in this mode where the FF indicates a delay flipflop and X designates exclusive OR gates.

A significant improvement in the degradation factor can be made by using the oversample mode when full bandwidth capability is not required. In this case the sample clock runs at 4 × B Hz. The autocorrelator contains an extra delay flipflop between each lag processor to permit one-half delay interval averaging. Figure 3 shows a block diagram of this configuration.

The 3 × 2 level autocorrelator requires a sampler that distinguishes three levels with properly selected thresholds and a two-level sampler identical to the previous system. The three-level signal can be thought of as containing levels −1, 0, and 1 and the two-level signal as containing −1 and 1. The autocorrelator counts the coincidence of delayed two-level samples with the −1 and 1's of the three-level present sample and inhibits the count when the present sample is zero.

Figure 4 shows an implementation of this scheme including the oversampling flipflops.

The combination of the (3 × 2) level sampling and oversampling yields an improvement in signal-to-noise ratio over the (2 × 2) level sampler operating at the Nyquist rate of 1.27. If this improvement were to be made up in observing time, a 60 percent increase would be required.

Further improvement of the signal-to-noise ratio can be achieved by using (3 × 3) level multiplication. However, the implementation of this feature results in one-half the number of channels. The oversample mode may still be used and results in a reduction in observing time by about a factor of 1.9 for equivalent signal-to-noise ratios relative to the (2 × 2) level system sampled at the Nyquist rate. In this case the multiplication matrix is as shown below.

$$\begin{array}{c|ccc} & -1 & 0 & 1 \\ \hline -1 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 1 \end{array} \quad \text{or} \quad \begin{array}{c|ccc} & -1 & 0 & 1 \\ \hline -1 & 2 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 2 \end{array}$$

In the first case an up-down counter could be used, but the borrow or negative counts create a large problem with respect to the word extension of the counter in memory. The second matrix requires different gating levels.

The implementation of the first matrix in this autocorrelator is accomplished by dividing each module in half and tallying separate parts of the matrix in each half such that the results can be assembled by the computer after the data are read out. In any case it is necessary to provide a delayed sample "zero" source. This source is used to inhibit the accumulators so that the zeros in the matrix are taken care of. The present sample line drives the first half of the unit while the complement of this sequence drives the second half and accumulates the −1's in the matrix. The proper autocorrelation function may be computed by subtracting the second set of accumulators from the first. Figure 5 shows a block diagram of the implementation of the logic including the gates for oversampling.

C. Bandwidth Doubling Modes

One of the features of the JPL–HAC autocorrelator is its ability to process signals having bandwidths greater than the maximum logic frequency. The machine is structured such that this can be done in two ways. The most obvious way is to operate the autocorrelator as a number of parallel units each accepting data from differing IF amplifiers and samplers. The system could be configured to drive each of the 8 units having 64 lags from separate IF amplifiers giving a total bandwidth of 80 MHz with a frequency resolution of 156.25 kHz. In this way resolution may be traded directly against bandwidth, permitting the following possible configurations (Table 3):

The second procedure that may be used, suggested by Art Shalloway, requires a single sampler operating at 2 or 4 times the 20-MHz sampling clock. The sequence of samples is separated into series containing every other or every fourth sample. For example, consider the simple bandwidth doubling mode. Let S be a series of samples numbered 1, 2, 3, 4, . . . which is broken into two new series containing the even and odd numbered samples. A contains 1, 3, 5, . . . and B contains 2, 4, 6, 8, ... The desired autocorrelation function $R_{ss}(n)$ is formed from the original sequence as shown in Eq. (1).

$$R_{ss}(n) = \sum_{k=1}^{N} (k,n+k) \tag{1}$$

where $(k,n + k)$ indicates the sample pairs used in the product and N is the total number of counts or clock pulses. Four correlation functions may be formed from the sequences A and B; these are R_{AA}, R_{BB}, R_{AB}, and R_{BA}. It is easy to show that R_{ss} can be formed by properly combining the four correlation functions by first separating Eq. (1) into even and odd summations as shown in Eq. (2).

$$R_{ss}(n) = \sum_{k=1,3,5}^{N} (k,n+k) + \sum_{k=2,4,6}^{N-1} (k,n+k) \tag{2}$$

Here, N has been assumed to be odd but results in no loss of generality as the upper limits of the sums can be interchanged if N is even. Now let ℓ be an even index 0, 2, 4, 6, . . . and m be an odd index 1, 3, 5, . . . and evaluate $R_{ss}(n)$ for each case.

$$R_{ss}(\ell) = \sum_{k=1,3,5}^{N} (k,\ell+k) + \sum_{k=2,4,6}^{N-1} (k,\ell+k)$$

$$= R_{AA}\frac{\ell}{2} + R_{BB}\frac{\ell}{2} \tag{3}$$

The odd lags may be evaluated in the same way resulting in Eq. (4).

$$R_{ss}(m) = \sum_{k=1,3,5}^{N} (k,m+k) + \sum_{k=2,4,6}^{N-1} (k,m+k)$$

$$= R_{AB}\left(\frac{m-1}{2}\right) + R_{BB}\left(\frac{m+1}{2}\right) \tag{4}$$

If 512 channels are available, 256 lags may be evaluated.

A similar analysis holds for sampling at 4 times the correlator rate and requires 16 correlation functions. This mode is more difficult to accomplish when only 8 units are available since each unit must be partitioned into half units. This may be accomplished by operating each unit in the "complex" mode. The complex mode permits each unit to use a second source for the delayed data stream to the second bank of 32 correlation channels. Thus each unit can form an autocorrelation function in the first bank if the data and multiplier lines contain identical data and a cross correlation against the data on the multiplier lines if the data line contains a different source in the second bank. If the data and multiplier lines on the first bank contain different data sources, two crosscorrelation functions may be formed in a single unit.

If the original sequence S is separated into four sequences *A*, *B*, *C*, and *D*, the following correlation functions can be set up in the units indicated. The second subscript is always a permutation beginning with the first which is always an autocorrelation function. This allows a simple scheme to be used to determine which part of memory contains the particular function desired to reconstruct R_{ss}. Note that 3 × 3 level processing is not possible since the correlator units are divided into the smallest banks that have been provided (Table 4).

D. Complex Autocorrelation Functions

The previous discussion indicates that crosscorrelation functions are easily accomplished since the delayed data lines and multiply lines are separate. The autocorrelation function of a complex function requires the formation of crosscorrelation functions between the real and imaginary parts of the complex signal.

If the autocorrelation function of a complex signal $s(t)$ is R_{ss}, then R_{ss} can be formed from the real and complex parts of s in the following manner:

$$R_{ss}(n) = R_{s_R s_R}(n) + R_{s_I s_I}(n) + j\left[R_{s_I s_R}(n) - R_{s_R s_I}(n)\right]$$

The autocorrelation must be partitioned into four sections each containing 2 units. Any modes of operation that can be implemented with 2 units can be used. Bandwidth doubling could be implemented by using half units.

E. Crosscorrelation Functions

Crosscorrelation functions of real data can be measured with any of the modes of Table 2 by providing separate data and multiplier samplers. Complex crosscorrelation functions may be measured in the same manner as complex autocorrelation functions by partitioning the correlator into 4 banks.

F. Autocorrelator Controller

The control unit for the autocorrelator system acts as a two-way interface between the autocorrelator and the PDP-11 control computer. It also greatly reduces the software support required to operate the system by keeping track of the overflows from the 24-bit counters internal to the autocorrelator. This is accomplished by supplying an overflow array address to the controller as well as a good-unit's control word that programs the sequence of unit addresses called by the controller to supply a display of the 24th bit of all lags. The PDP-11 memory is incremented by direct memory access when respective overflows occur. This procedure also frees the PDP-11 for other data reduction chores during the "run" cycle.

The end of the "run" cycle is indicated by a controller-generated interrupt that may be used to inform the PDP-11 to unload the data or to continue to some other task. A fairly complete description of the operation of the controller is given in Section II on commands and programming. The controller is capable of establishing all possible operation modes of the autocorrelator.

G. Interim Sampler Module

The present sampling unit consists of a two-level sampler having a digital direct current (dc) removal system that feeds back a reference signal to a comparator in order to maintain the average number of 1's and 0's equal (Ref. 2). With the present configuration, only the first two autocorrelator units may be driven from the sampler. The sampler is also wired to permit only autocorrelation; that is, the data and multiplier lines are supplied with the same signal. The sampler unit also distributes the master clock, the run, and extended run signals to the first two units. This configuration permits bypassing unit 1 should it fail to operate properly. All other units presently may be driven from unit $N - 1$ or $N - 2$. No provisions are available for bandwidth doubling or multilevel sampling at the present time.

II. Control, Operation, and Test Software for the JPL–HAC Autocorrelator

This section describes the procedure for controlling the JPL–HAC autocorrelator via the PDP-11/20 computer. Descriptions of the controller registers and required function codes are given with examples shown in PAL-11 code. The appendix section shows an example of a subroutine written in the Sigma 5 metasymbol code for the PDP-11 cross assembler (Ref. 3). Examples of the register loading, data readout sequence, and run cycle commands are given.

A. Autocorrelator Controller Registers

Operation of the autocorrelator requires that the controller be properly initialized. This consists of loading the first six internal registers of the controller from a list called the control register vector (CRV). Figure 6 shows the configuration for the CRV. The first 5 registers (0 through 4) contain 16-bit words while the 6th register contains the most significant byte (8 bits) of the 40-bit zero lag counter. The zero lag counter must be loaded with the 1's complement of the true number of counts desired following the commencement of the run command and must be reset prior to each run command if a precise run length is desired. Registers 0, 1, and 2 need not be reset. Control register 1 contains the good-units word. This register may have a maximum of 9 bits set in the locations 0 through 8. Each bit set corresponds to an operating unit. Thus, if the first four units are to be operated, 17_8 would be transferred to this register. Unit 2 could be bypassed by sending 25_8 to this register. In this case, more advanced software could establish that unit 1 is to be driven from the data sampler, unit 3 from unit $N - 2$, and units 4 and 5 from units $N - 1$.

Control register 2 contains the starting address of the array in which overflow incrementing is to begin for each channel of all operating units. The overflow array provides a 16-bit extension to the 24-bit counters internal to the autocorrelator. Normally no more than 8 units will be operated so that the array need contain only 1000_8 words. Control register 5, which contains the most significant 8 bits of the zero lag counter, will return zeroes in the upper byte of the word when read back irrespective of what is written there. Since the controller does not respond to byte transfers from the PDP-11, a MOVB instruction may not be used to transfer data to this register. A MOVB used to read CDR-6 will always return the lower byte and will respond only to the even address boundary 164002_8.

Registers 6 and 7 are used for data communication between the autocorrelator and the control computer (PDP-11/20). The selection of a particular register is made by sending the register number to the status and control register (SCR) located at address 164000_8. This establishes the multiplexing of the 6 CRV words into the proper controller data register (CDR) located at 164002_8. As an example, loading the interrupt vector address is accomplished by the following instructions:

```
SCR = 164000
CDR = 164002

MOV #0,SCR        select register 0
MOV #270,CDR      write interrupt
                   vector address
                   in register 0
CMP CDR,#270      check register 0
BNE ERROR         go to error
                   routine
```

Each autocorrelator unit contains a bus function register (BFR) consisting of 12 bits that establish the signal source configuration and operation mode. These data are passed through CDR #6 by selecting register 6 along with a unit address to which the data are to be transmitted. Unit addresses 1 through 11_8 are legal and address 17_8 addresses all units. The "all units" address may be used to set identical data in the BFRs of all units but obviously cannot be used to read data from all units.

B. Controller Status and Control Register Configuration

In order to load the BFRs, the status and control register must be properly programmed. The bit configuration of this register is shown in Fig. 7.

Bits 0–2 point to the controller data register.

Bits 3–4 normally contain zeroes, but are used in the data readout sequence.

A one placed in bit 3 will lock out or protect the data register from further write commands to location 164002_8.

Bits 5–6 specify the interrupt priority level of the correlator when the interrupt is generated at the end of the run.

Bits 7–10 specify the unit address as discussed earlier.

Bits 11–12 correspond to the bus control function codes of the autocorrelator.

00 = connect

01 = load bus function register

10 = data reset

11 = initialize

Bit 13 is a computer operated run control.

1 = run

0 = stop

Bit 14 is a run status flag.

Bit 15 indicates a time out if a correlator unit does not respond within 10 μs.

C. Bus Function Register Configuration

Figure 8 shows the Bus Function Register configuration internal to each autocorrelator unit.

The bit assignments for the Bus Function Register are as follows:

Bits 0–1	Signal source	0 — from stage N − 1
		1 — from stage N − 2
		2 — from data sampler
		3 — from auxiliary unit
Bits 2–3	Data type	0 — real two level
		1 — real three level
		2 — complex two level
		3 — complex three level
Bits 4–5	Multiplier type	0 — 2 level
		1 — 3 level
		2 — 4 level
		3 — 5 level
Bit 6	Oversample	0 — Nyquist rate
		1 — 2x Nyquist rate
Bit 7	Overflow read	0 — data read mode
		1 — overflow read mode
Bit 8	Low-speed clock	0 — high-speed clock
		1 — low-speed clock
Bit 9–11	Test mode	0 — operate
		1 — disable data bus
		2 — bus transmit 1's
		3 — bus transmit 0's
		4 — memory load 1's
		5 — memory load 0's
		6 — spare
		7 — fault indicator on

Suppose 8 units are to be operated in the simplest mode (real-two level data, two level multiply). #13606_8 sent to the SCR selects the data transfer register, CDR-6, selects write mode on all units, and prepares for the loading of the bus function register. #200_8 may be sent to the CDR to specify that all units are to be driven from unit N − 1, and the overflow read mode is set by bit 8. Since unit 1 has no source unit, the data and run command signals must be sent from the controller and data sampler. This is accomplished by sending #10206_8 to the SCR followed by #202_8 to the CDR.

D. Initialize and Data Reset

The initialize command resets the entire correlator and generates a data reset which clears the low order 8-bit counters and prepares the 16-bit memory for clearing when the run command is sent. Approximately 4 μs is required to clear the memory. The initialize also clears the Bus Function Register, which must be reprogrammed prior to the run command. The data reset command clears the low order 8-bit counters and prepares the 16-bit counters for clearing upon sending the run command. The data reset signal also resets the memory address sequencer to zero. Since the data in the memory are not destroyed until the run is enabled, the memory can be reread repeatedly. However, the low order 8 bits will read zero for one cycle and a checkerboard test pattern for all successive read cycles. The initialize signal is also generated during the power-up cycle and insures that all counters and control memory devices are in the initialized state. These commands may be sent to all units simultaneously by using the all-units address code. The codes follow:

```
MOV    #3606,SCR    Initialize all units
CLR      CDR
MOV    #7606,SCR    Data reset all units
CLR      CDR
```

E. Run Control

Presently, the autocorrelator may be set in the run mode by sending a #34000_8 to the SCR. This enables the run bit and sets the connect code so that subsequent overflow read requests generated by the controller are recognized by any unit that is addressed. The run command may be removed before the zero lag counter has finished and the interrupt is fired. This feature permits the aborting of a long run that is discovered to be defective for any reason, but because the zero lag counter is halted, the overflow scan cycle is prematurely terminated. Therefore, some overflows may not have been tallied, and reading the data from the autocorrelator may not be useful.

Because the run control is computer-actuated, the exact starting time is not exactly known. Future modifications will permit the computer to generate a run enable signal, which will permit an external clock to generate the run command at a precise time.

F. Data Read Mode

The interrupt service routine, which is initiated once the countdown of the zero lag counter is completed, generally initiates a data unloading sequence. The low-order bytes and medium-order words are generally loaded into separate arrays, as this can be accomplished more quickly than the packing of the data into 40-bit words. The packing could be done once the next run is initiated if time permits. The read request sequence requires that the autocorrelator be in the connect mode and that a read request be generated by the controller. The order of reading CDR 6 and 7 is unimportant; however, bit 5 of the SCR must be set during the first read and reset to zero during the second read. The toggling of bit 5 actually establishes the read request to the autocorrelator and increments the memory address. The autocorrelator is first set in the data read mode as follows:

```
MOV     #13606,SCR     Set data read mode

CLR       CDR          On all units
```

It is convenient to take advantage of the autoincrement addressing modes of the PDP-11 while unloading the data. If MOWA and LOBA are the first address locations of the medium-order word array and the low-order byte array, then a simple program to unload unit 1 might be written as:

```
MOWA = 3000₈

LOBA = 3200₈

       MOV     LOBA,R1       Put array addresses
       MOV     MOWA,R2       in registers 1 and 2
       MOV     #101,RO       Initialize loop
                             counter

LOOP   DEC     RO            Decrement loop
                             index

       BEQ     EXIT          Go to exit if index
                             is zero send
                             connect, lockout
                             write mode

       MOV     #14236,SCR    Send read request
                             and select CDR 6

       MOVB    CDR,(R1)+     Read CDR 6

       MOV     #14217,SCR    Lock out write
                             mode send connect
                             toggle bit 5
                             select CDR 7

       MOV     CDR,(R2)+     Read CDR 7

       BR      LOOP          Continue loop

EXIT - - - - -
```

The medium-order words now fill 100_8 words and the low-order bytes fill 40_8 words, beginning at these respective addresses. The overflow array also fills 100_8 words. These three arrays comprise the entire data output. If it is known that the zero lag counter contains less than a 16-bit count, the MOV CDR,(R2)+ instruction may be replaced with a MOVB to cut the MOWA in half. In the case of very long runs, the least significant bits do not have to be read, but the toggle of bit 5 must be sent to the SCR.

The appendix shows an example of a subprogram to unload *N* good units as specified by the good units word (GUW).

G. Test Modes

The autocorrelator may be operated in five test modes. These permit the computer to execute a number of diagnostic tests to determine if a particular unit is operating properly. These tests are not exhaustive, and complete testing requires the operation of the autocorrelator with certain test signals for which the exact autocorrelation function is known. These test signals are exact square waves counted down digitally from the master clock and are available behind the data sampler module.

1. Test mode 1 (disable data bus). Test mode 1 may be established by loading the BFR with 1000_8. This mode disables the tristate bus driver, and subsequently reads from the data bus should return 177777_8 through the CDR.

2. Test mode 2 (bus transmit 1's). Test mode 2 is established by loading the BFR with 2000_8. This mode disables the tristate drivers of the memory multiplexer bus that feeds the output bus. A series of 64 read requests will cycle through the entire multiplex sequence, insuring that the multiplex bus is free in all address modes if 177777_8 is returned through the CDR. The lower 8 bits also transmit 1's through the data bus in this mode.

3. Test mode 3 (bus transmit 0's). Test mode 3 transmits zeroes to the output bus by disabling the memory output lines. These lines are inverted and passed through the multiplex bus. Therefore, the inverters as well as the memory disable may be tested. The low-order 8 bits also will indicate zeroes when read, since test modes 1 and 3 generate a data reset signal that clears the 8-bit counters. This test mode assures that the 8-bit counters clear if all 64 reads indicate zeroes.

4. Test mode 4,5 (memory transmit 1's, memory transmit 0's). Test modes 4 and 5 are somewhat more difficult

to set up and require that a large amount of the autocorrelator be operating properly. In particular, the memory scan circuits must be operating properly. Since the memory scanning does not operate unless the autocorrelator is in the "run" mode, a short run cycle must be executed to cause the memories to clear all addresses. This requires about 4 μs. In test mode 4, the memories are cleared and decremented so that all 1's are presented to the multiplex bus. Test mode 5 is identical except that the decrement signal is not generated. The low-order 8-bit data bus will read the contents of the 8-bit counters. A data reset command will clear these counters and insure that zeroes will be read in either test mode 4 or 5. The ability of these counters to load 1's and 0's may be tested by repeating the read sequence. During the first read sequence the data latches of these counters are filled with an alternating 1-0 pattern so that an 8 $\times$ 64 checkerboard pattern exists. During the second read cycle, the 16-bit memory will repeat the original data, as these data are not destroyed during the readout. The 8 low-order bits will alternately read 125_8 and 252_8.

In order not to have to set up the interrupt instructions, it is convenient to use the manual run mode to provide the run signal. Clearing the zero lag counter will insure that no interrupt can happen for roughly 8 hours. An example of a program to check test mode 4 is shown below.

```
MOV #3, SCR              Clear zero lag counter
CLR CDR
MOV #4, SCR
CLR CDR
MOV #5, SCR
CLR CDR
MOV #206, SCR            Initialize unit 1
CLR CDR
MOV #10206, SCR          Select load bus function
MOV #4002, CDR           Select test mode 4
                         and drive unit from
                         data sampler
MOV #34000, SCR          Run
NOP                      Wait 4 microseconds
NOP
NOP
CLR SCR                  Stop run
MOV #1, GUW              Set good units word
JSR R5, UNLDAC           Unload unit 1
WORD GUW, MOWA, LOBA
```

Arrays located at MOWA and LOBA now contain the results of the test. The 16-bit MOWA should contain 177777_8 in all 64 words, and the 8-bit LOBA should contain 1's in all 64 locations. A second call to the unload routine will cause the checkerboard pattern to be loaded into the LOBA.

Test mode 6 is unimplemented, and test mode 7 causes the fault lamp to operate.

H. Reading the BFR

The first 8 bits of the Bus Function Register may be read on the 8-bit data bus when the autocorrelator is in the overflow read mode. In order to accomplish this, the autocorrelator must be placed in the run mode long enough for the controller to issue an overflow read request. A countdown from 2^{24} in the zero lag counter is adequate for this purpose. It is not possible to know which unit was read last at the finish of the run cycle. Therefore, this test requires that the good-units word, GUW, contain only one set bit; i.e., only one unit is operated at a time. As a result, the reading of the BFR is generally useful only for test purposes and not for verifying the operational configuration.

References

1. Bowers, F. K., and Klinger, R. J., *Quantization Noise of Correlation Spectrometers*, Department of Electrical Engineering, University of British Columbia, Vancouver, British Columbia.

2. Brokl, S. S., and Hurd, W. J., "Digital DC Offset Compensation of Analog-to-Digital Converters," in *The Deep Space Network*, Technical Report 32-1526, Vol. XVII, Jet Propulsion Laboratory, Pasadena, Calif., Oct. 15, 1973.

3. Erickson, D. E., "The SAPDP Program Set for Sigma 5 Assembly," in *The Deep Space Network*, Technical Report 32-1526, Vol. VII, Jet Propulsion Laboratory, Pasadena, Calif., Feb. 15, 1972.

Table 1. Degradation factors (Ref. 1)

Levels	Nyquist	2 × Nyquist	3 × Nyquist
2 × 2	1.57	1.35	1.29
2 × 3	1.39	1.24	1.19
3 × 3	1.23	1.13	1.11
3 × 5	1.15	1.09	1.07
4 × 4	1.13	1.07	1.06

Table 2. Operation modes

Signal bandwidth, MHz	Signal levels	Sampling rate	Channels available
40	2 × 2, 3 × 2	Nyquist	128
20	2 × 2, 3 × 2	Nyquist	256
20	3 × 3	Nyquist	128
10	2 × 2, 3 × 2	Nyquist	512
10	3 × 3	Nyquist	256
5	2 × 2, 3 × 2	Nyquist	512
5	2 × 2, 3 × 2	2 × Nyquist	512
5	3 × 3	Nyquist	256
5	3 × 3	2 × Nyquist	256

Table 3. Bandwidth vs resolution using multiple IF amplifiers

Total BW, MHz	Number of IFA	Number of lags/IF	Resolution, kHz
80	8	64	156.25
40	4	128	78.125
30	3	192	52.0833[a]
20	2	256	39.0625
10	1	512	19.53125

[a]Using spare unit to yield 3 banks of 3 units.

Table 4. Signal drive sources for bandwidth quadrupling mode

Unit	R_1	R_2	Mul	D_1	D_2
1	R_{AA}	R_{AB}	A	A	B
2	R_{AC}	R_{AD}	A	C	D
3	R_{BB}	R_{BC}	B	B	C
4	R_{BD}	R_{BA}	B	D	A
5	R_{CC}	R_{CD}	C	C	D
6	R_{CA}	R_{CB}	C	A	B
7	R_{DD}	R_{DA}	D	D	A
8	R_{DB}	R_{DC}	D	B	C

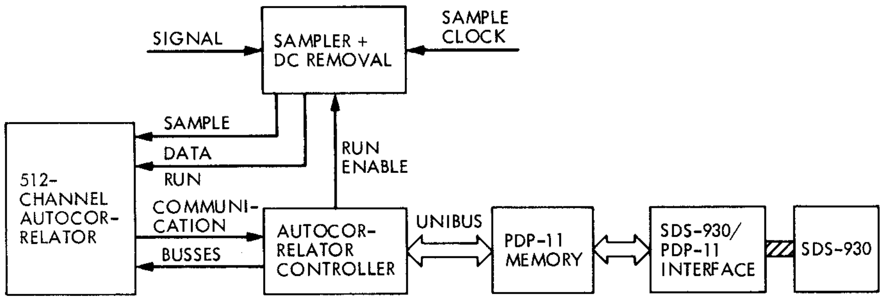


Fig. 1. Block diagram of the autocorrelator system

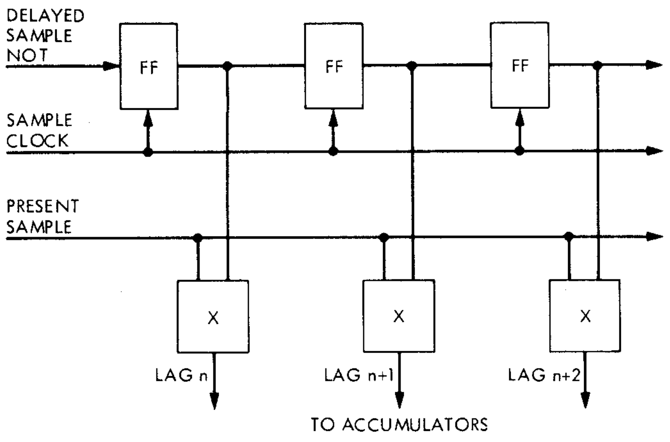


Fig. 2. Two-level autocorrelator at Nyquist rate

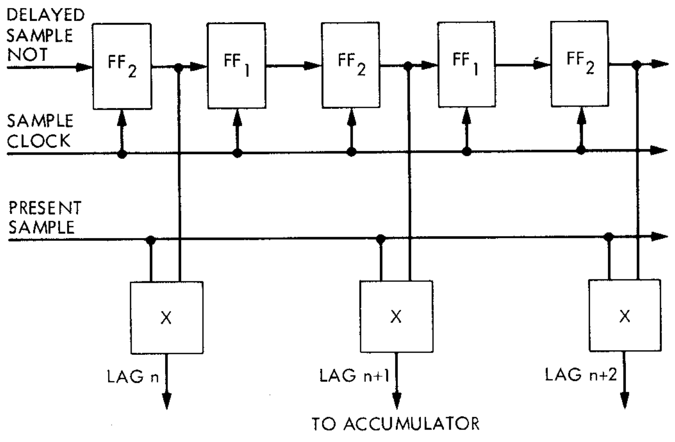


Fig. 3. Oversample mode in 2 × 2 levels

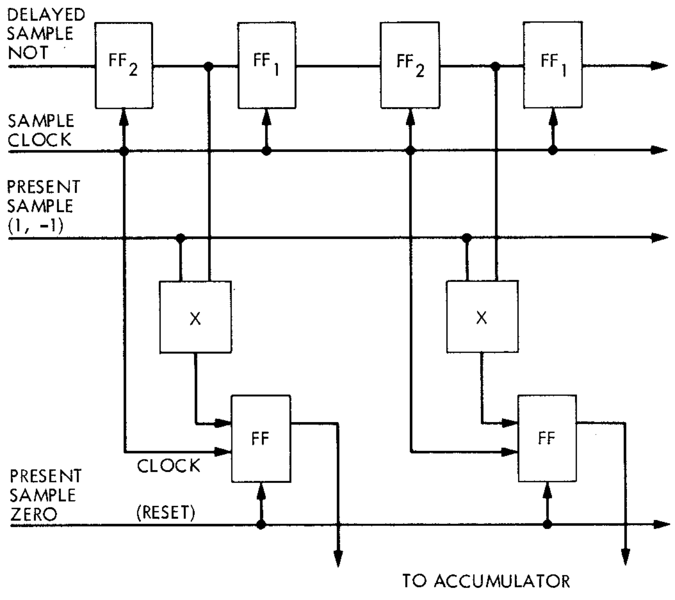


Fig. 4. Autocorrelator with oversampling, (3 × 2) level

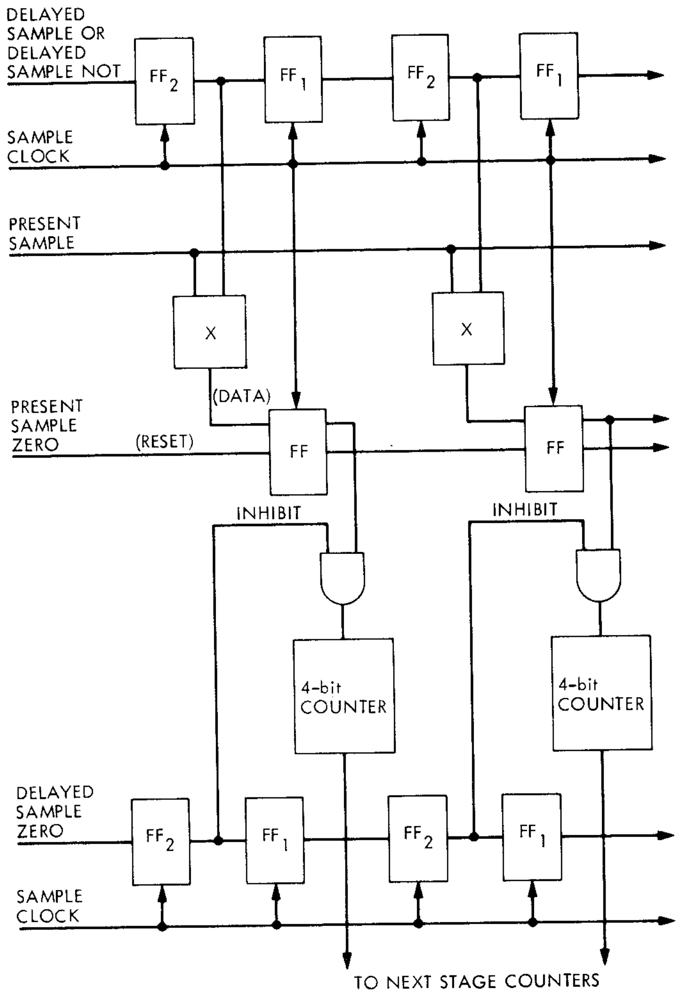


Fig. 5. Autocorrelator with oversampling, (3 × 3) level

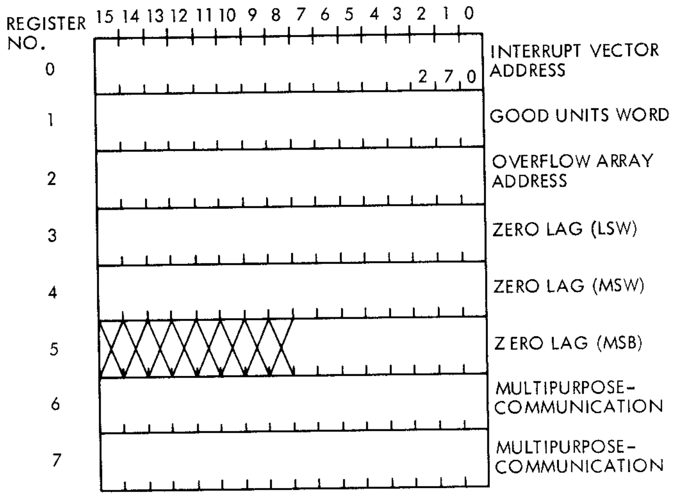


Fig. 6. Control register vector

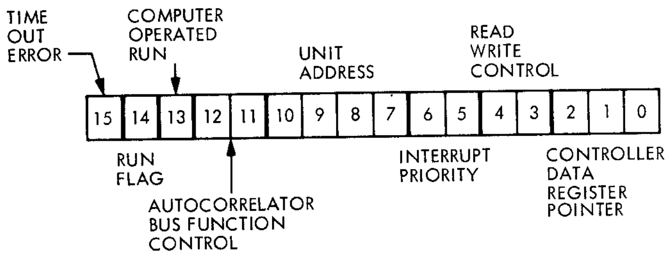


Fig. 7. Status and control register configuration

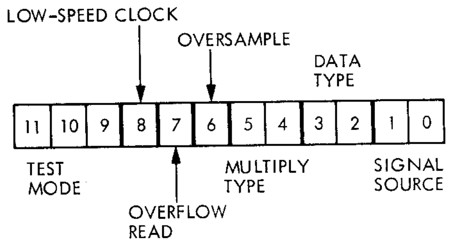


Fig. 8. Bus function register configuration

Appendix

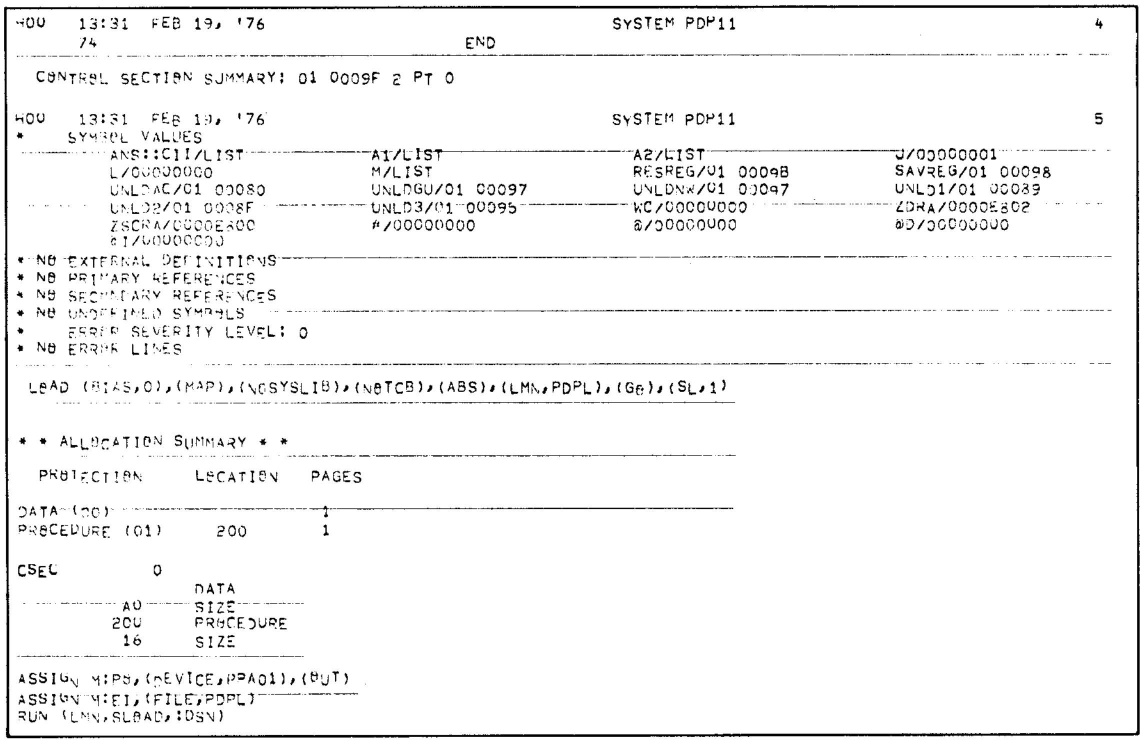
Subroutine for unloading N units

```
 13:31 FEB 19,'76 ID=00OF-F00
 JOB AUTOCOR,JURGENS
METASYM SI,LO,SO

HOU    13:31   FEB 19, '76                                  SYSTEM PDP11                             1
        01 00080                           ORG,1    O'1000'
        01 00080

HOU    13:31   FEB 19, '76                                  SYSTEM PDP11                             2
                                    SYSTEM   PDP11
                                  *  SUBROUTINE TO UNLOAD HAC-JPL AUTOCORRELATOR
                                  *  CALLING SEQUENCE:   JSR   5, UNLDAC
                                  *                      WORD  GUW,MOWA,LOBA
                                  *
                                  *  R.F.JURGENS  7   JAN 76
                                  *
        01 00080      09F7005C A   UNLDAC JSR       7,SAVREG
        01 00081      15F71786 N          MOV       (#,O'13606'),ZSCRA  SET DATA READ MODE
                      E5F6
        01 00082 2    0A37E5F4 N          CLR       ZDRA
        01 00083 2    1777004A A          MOV       *(5,@),UNLDGU      PICK UP GOOD UNITS WORD
        01 00084 2    1742     A          MOV      *(5,@),2            PICK UP MOWA
        01 00085      1741     A          MOV      *(5,@),1            PICK UP LOBA
                                  *  INITIALIZE READ REQUEST CODES
        01 00085 2    15C3181E A          MOV       (#,O'14036'),3     UNIT 0 LOB READ REQUENT CODE
        01 00086 2    15C4180F A          MOV       (#,O'14017'),4     UNIT 0 MOW READ REQUEST CODE
                                  *  UNIT LOOP
        01 00087 2    15F7000A A          MOV       (#,10),UNLDNW      INITIALIZE UINT COUNTER
                      003A
        01 00089      0AF70036 A   UNLD1  DEC       UNLDNW             DEC UNIT COUNTER
        01 0008A      0316     A          BEQ       UNLD3
        01 0008A 2    65C30080 A          ADD       (#,O'200'),3       INCREMENT UNIT ADDRESS CODES
        01 0008B 2    65C40080 A          ADD       (#,O'200'),4
        01 0008C 2    0C370026 A          ROR       UNLDGU             ROTATE GOOD UNIT BIT INTO CARRY
        01 0008D 2    86F6     A          BCC       UNLD1
                                  *  LAG LOOP
        01 0008E      15C00041 A          MOV       (#,65),0           INITIALIZE LAG COUNTER
        01 0008F      0AC0     A   UNLD2  DEC       0                  LAG COUNT LOOP
        01 0008F 2    03F2     A          BEQ       UNLD1
        01 00090      10F7E58C N          MOV       3,ZSCRA            READ LOB
        01 00091      1001E58A N          MOV       ZDRA,(1,@)
        01 00092      1137E584 N          MOV       4,ZSCRA            READ MOW
        01 00093      1002E582 N          MOV       ZDRA,(2,@)
        01 00094      00A0     A          NOP
        01 00094 2    00A0     A          NOP
        01 00095      01F3     A          BR        UNLD2

HOU    13:31  FEB 19, '76                                   SYSTEM PDP11                             3
        01 00095 2  09F70014 A   UNLD3    JSR       7,RESREG
        01 00096 2  0085     A            RTS       5
        01 00097                 UNLDGU RES,2       1                  GOOD UNITS SCANNER WORD
        01 00097 2               UNLDNW RES,2       1                  NUMBER OF UNITS INDEX
              0000E800           ZSCRA  EQU        O'164000'
              0000E802           ZDRA   EQU        O'164002'
                                 *
                                 *
                                 *  SUBROUTINE SAVREG
                                 *  CALLING SEQUENCE  JSR  7,SEVREG
                                 *  SUBROUTINE TO SAVE GENERAL REGISTERS ON STACK WHEN R5 IS USED AS
                                 *  A SUBROUTINE LINKAGF REGISTER.   RO TO R4 ARE SAVED.
                                 *  CALL TO SAVEREG MUST BE FIRST INSTRUCTION IN SUBROUTINE TO INSURE
                                 *  THAT NO OTHER DATA IS PLACED ON THE STACK.
                                 *
        01 00098    1126     A   SAVREG MOV       4,(@,6)           PUT REGISTERS 0 TO 4 ON STACK
        01 00098 2  10E6     A          MOV       3,(@,6)
        01 00099    10A6     A          MOV       2,(@,6)
        01 00099 2  1066     A          MOV       1,(@,6)
        01 0009A    1026     A          MOV       0,(@,6)
        01 0009A 2  007E000A A          JMP       *(10,6)           RETURN TO SUBROUTINE
                                 *
                                 *
                                 *  SUBROUTINE RESREG
                                 *  CALLING SEQUENCE:    JSR  7,RESREG
                                 *  SUBROUINTE TO RESTORE REGISTERS AT EXIT OF SUBROUTINE  WHICH CALLED
                                 *  SAVREG AS FIRST INSTURCTION.   RESREG MUST BE CALLED BEFORE THE
                                 *  RTS 5  AT THE EXIT OF THE SUBROUTINE
                                 *
        01 0009B 2  1536000A A   RESREG MOV      (6,@),(10,6)       MODIFY LINKAGE RETURN ADDRESS
        01 0009C 2  158C     A          MOV,     (6,@),0            RESTORE REGISTERS 0 TO 4
        01 0009D    1581     A          MOV      (6,@),1
        01 0009D 2  1582     A          MOV      (6,@),2
        01 0009E    1583     A          MOV      (6,@),3
        01 0009E 2  1584     A          MOV      (6,@),4
        01 0009F    005E     A          JMP      *(6,@)             RETURN TO NEW LINKAGE ADDRESS
                                 *
```



```
H00    13:31   FEB 19, '76                                          SYSTEM PDP11                                      4
       74                                          END

  CONTROL SECTION SUMMARY: 01 0009F 2 PT 0

H00    13:31   FEB 19, '76                                          SYSTEM PDP11                                      5
*      SYMBOL VALUES
          ANS::CII/LIST               A1/LIST                       A2/LIST                       J/00000001
          L/00000000                  M/LIST                        RESREG/01 00098               SAVREG/01 00098
          UNLDAC/01 00080             UNLDGU/01 00097               UNLDNW/01 00097               UNLD1/01 00089
          UNLD2/01 0008F              UNLD3/01 00095                WC/00000000                   ZDRA/0000E802
          ZSCRA/0000E800              #/00000000                    @/00000000                    @D/00000000
          @I/00000000
* NO EXTERNAL DEFINITIONS
* NO PRIMARY REFERENCES
* NO SECONDARY REFERENCES
* NO UNDEFINED SYMBOLS
*      ERROR SEVERITY LEVEL: 0
* NO ERROR LINES

 LOAD (BIAS,0),(MAP),(NOSYSLIB),(NOTCB),(ABS),(LMN,PDPL),(GO),(SL,1)


* * ALLOCATION SUMMARY * *

  PROTECTION      LOCATION    PAGES

DATA (00)                       1
PROCEDURE (01)      200         1

CSEC            0
                    DATA
            A0      SIZE
           200      PROCEDURE
            16      SIZE

ASSIGN M:PO,(DEVICE,PPA01),(OUT)
ASSIGN M:EI,(FILE,PDPL)
RUN (LMN,SLOAD,:DSN)
```