Digital Multiplexing of Analog Data in a Microprocessor
Controlled Data Acquisition System

RODER A. CLIFF

Abstract—An analog data acquisition system typically consists of an analog multiplexer followed by an analog to digital (A/D) converter. An alternative configuration uses multiple comparators (one per input) followed by a digital multiplexer. If the control functions for A/D conversion are incorporated into a microprocessor, several potentially interesting data acquisition and conversion strategies are available. At first thought, an interrupt-driven conversion process which simultaneously searches all inputs in parallel for a level match appears attractive. However, analysis shows that a sequential conversion of the inputs, one at a time, using the successive approximation algorithm, is usually superior.

I. INTRODUCTION

Analog data acquisition is a function of many microprocessor systems. Typically the part of the system which does the analog data acquisition uses an analog multiplexer which (under microprocessor control) selects one of the analog inputs and transmits it to an analog to digital (A/D) converter. The selected analog input is digitized by the A/D converter and the digitized value of the analog input is transmitted to the microprocessor [1].

This method of analog data acquisition is perfectly acceptable for most applications. There are, however, some problems associated with the analog multiplexer, which normally consists of an array of analog semiconductor switches [2]. It is the purpose of this correspondence to explore the ramifications of doing the multiplexing function in the digital domain and of using the microprocessor to replace the hardwired control logic which resides in the typical A/D converter.

II. ELIMINATION OF ANALOG MULTIPLEXOR

A/D conversion can be accomplished in many ways [1]. At the black box level, an A/D converter partitions the analog input space into n contiguous disjoint subspaces. It further makes a decision as to which of these subspaces the current value of the input variable resides in. It then outputs a digital code which uniquely identifies that subspace.

If the input signal could change significantly in the time required to do the A/D conversion, then it would be necessary to include a sample-and-hold circuit to freeze the signal level applied to the A/D converter. However, if the signal cannot change significantly during the conversion time (a thermistor output for instance), then the sample-and-hold is not required [3].

One way to eliminate analog multiplexing is to employ a separate A/D converter for each analog input [3]. Indeed for some systems (for instance if there are relatively few wide-bandwidth inputs) this may be the most viable approach. However, for systems with large numbers of narrow-bandwidth inputs (such as spacecraft housekeeping data) there is a more practical method.

A frequently used type of A/D converter functions by repeatedly comparing its analog input to a sequence of analog levels which are generated by a D/A converter [1]. At each step of this process, the output of the comparison function is binary. Therefore, if only the comparison function is replicated, the multiplexing can be done in the digital domain.

This technique is particularly attractive for narrow-bandwidth inputs which do not require a sample-and-hold because then the only hardware which need be replicated per input is the comparators. In fact, the comparators can be located remotely with the analog sources. If this is done, the operational amplifiers which are typically used to buffer analog sources are no longer required. They are supplanted by an equal number of comparators.

III. ELIMINATING CONTROL LOGIC

Hardwired logic is typically used to control the D/A converter which generates the test levels which go to the comparator(s) [1]. If the speed requirements of conversion are not particularly stringent and the microprocessor is not overly busy on other tasks, the control logic function can be subsumed into the microprocessor. Aside from eliminating unnecessary hardware, another advantage is obtained—flexible control of the A/D conversion process.

Suppose, for instance, that a large number of inputs need only be quantized to 4 bits, whereas several require quantization to 12 or more bits. The microprocessor can handle this by appropriate choice of the levels it commands of the D/A converter. The result can be a major saving in conversion time.

Furthermore, inputs of varying polarity and dynamic range can be accommodated. Conversion time can also be reduced for narrow-bandwidth signals by searching first those levels close to the last recorded value.

IV. CONVERSION STRATEGIES

Numerous conversion strategies are possible with the microprocessor in full control of the conversion process. Aside from the adaptive strategies alluded to above, there are several other ways to convert n inputs to b bits each. Two of these possibilities will now be explored.

It would seem that there might be some strategy which would allow simultaneous searching of all the inputs in parallel, and indeed there is. The entire set of possible input levels can be exhaustively searched (e.g., from lowest to highest), and an interrupt to the microprocessor can be generated whenever a correct match occurs.

The hardware to accomplish this is shown in Fig. 1. The function of the comparator output gates is to disable the output of each comparator after it has caused an interrupt. Once the output of a comparator becomes true, it will remain so until the end of that search cycle. This would interfere with the processing of other comparator outputs if the comparator gates were not used. The details of the comparator output gate are given in Fig. 2.

Now let us estimate the time required to do a conversion using this technique. First, consider the case where only one input is used. In this case the time required to get the test level to the comparator inputs is negligible. Then, for each interrupt (potentially n of them) n comparator outputs must be tested. Thus, the time required for conversion varies roughly as 2^b + n^2. This algorithm was programmed using an arbitrarily chosen instruction set and it was found that the execution time (number of instructions) was

\[ t = 6(2^b - 1) + n(4n + 6). \]

(1)

Let us now estimate the time required for the usual successive approximation conversion algorithm. This type of conversion must be done one input at a time, since at each step of the process the next test level depends on the result of the comparison which was just completed. The time to convert one input is proportional to the number of bits, b. Therefore, the
time to convert $n$ inputs will be roughly proportional to $n b$. The successive approximation algorithm was also programmed (using the same instruction set as before) and the execution time was found to be

$$t = n(9b + 5).$$

The superiority of the successive approximation algorithm under almost any conceivable conditions is perhaps best shown by plotting (1) and (2). This has been done in Fig. 3 for the case of 4 bit conversion. For larger numbers of bits, the superiority of the successive approximation algorithm becomes increasingly evident. The plot covers the only region where the interrupt driven system even approaches the performance of the binary search method.

It would in principle be possible to do a vectored interrupt such that each time an interrupt occurred it would not be necessary to exhaustively search to see which input caused the interrupt. Unfortunately, for nontrivial numbers of inputs an undesirable amount of hardware would be required to accomplish this. Furthermore, the possible occurrence of simultaneous interrupts from multiple inputs makes the problem even thornier. For these reasons vectored interrupts are excluded from further consideration.

V. CONCLUSION

The advantages of using a digital multiplexer are that, contrary to an analog multiplexer, it does not degrade the fidelity of the conversion process. Conversion time is not increased if a successive approximation algorithm is used in the A/D conversion process. The required replication of comparators need not be a system disadvantage, because they can replace op-erational amplifiers which would otherwise be used as analog output buffers.

Given that a microprocessor is used to directly control the conversion process, several conversion strategies have been analyzed. Although an interrupt driven approach might, at first glance, seem to be attractive, analysis shows that successive approximation is a superior strategy. In special cases such as narrow-bandwidth inputs, or inputs requiring differing degrees of resolution, tailoring the search to the input parameters could be of benefit. This aspect of the situation has not yet been analyzed in detail.
dundancy (TMR) is discussed when failures exist not only in any single module but also in any two or three modules at a time. The optimization is performed periodically so that additional transient failures can be tolerated. The ordinary TMR under dependent-failures, a new fault-tolerant microcomputer system is proposed where a program is executed three times by three CPU's. For the purpose of eliminating the effect of dependent-failures, the interrelationship of failures between triplicated modules is modeled as shown in Fig. 2. The parameters are expressed as follows where it is assumed that the reliability of memory and I/O subsystems can be perfectly determined by another technique such as error-correcting codes.

The system consists of three CPU's and memory containing CPU are connected to memory or I/O ports through majority voters. Resynchronization of triplicated CPU's is started periodically by the interrupt of the timer output.

In this paper, a theoretical consideration is done on the reliability of a microcomputer system with dependent-failures. In order to evaluate the reliability of a microcomputer system, effective reliability is defined which is the reliability at the time when a certain quantity of processing has been completed. If dependent-failures occur frequently, it can be made clear that effective reliability cannot be improved more than a certain value. For the purpose of eliminating the effect of dependent-failures, a new fault-tolerant microcomputer system is proposed where the same program is not executed in any two processors simultaneously.

II. RELIABILITY ANALYSIS OF THE TMR NETWORK UNDER DEPENDENT-FAILURES

Fig. 1 shows a typical microcomputer system based on TMR. The system consists of three CPU's and memory containing programs and data. Data, address, and control outputs of each CPU are connected to memory or I/O ports through majority voters. Resynchronization of triplicated CPU's is started periodically by the interrupt of the timer output.

For simplicity, the following assumptions are made.

1) The reliability of memory and I/O subsystems can be perfect by another technique such as error-correcting codes.

2) Only transient failures occur in the system.

In order to define the parameters which express the degree of dependent-failures, the interrelationship of failures between triplicated modules is modeled as shown in Fig. 2. The parameters are defined as follows where it is assumed that the relationships of failures between any two or three modules are symmetric:

\[ \begin{align*}
\lambda_0 & \quad \text{failure rate of a single module;} \\
\lambda_V & \quad \text{failure rate of total voters;} \\
\lambda_3 & \quad \text{failure rate for simultaneous failure in all the three modules;} \\
\lambda_0 + 2\lambda_0 & \quad \text{failure rate for simultaneous failure in exactly two of three modules;} \\
\lambda_0 + 3\lambda_0 & \quad \text{failure rate for failure in exactly one of three modules;} \\
\lambda_0 & \quad \text{failure rate for simultaneous failure in both of two modules;} \\
\lambda_0 & \quad \text{failure rate for exactly one of two modules.}
\end{align*} \]

Now, let \( \lambda_2 \) of \( s = \lambda_2 \) and \( \lambda_3 \) of \( s = \lambda_3 \), so the other parameters are expressed as (1):

\[ \begin{align*}
\lambda_1 &= 3\lambda_0 - 2\lambda_2 - 3\lambda_3 \\
\lambda_2 &= 2\lambda_0 - 2\lambda_2 \quad \text{of 2} \\
\lambda_3 &= \lambda_2 / 3 + \lambda_3.
\end{align*} \]