MULTIPLEXING HIGH SPEED PCM DATA WITH PREPROCESSORS

James Willis
Systems Development Section
Physical Science Laboratory
Box 3548
New Mexico State University
Las Cruces, NM 88003

ABSTRACT

This paper deals with the use of preprocessors to reduce loading on real-time computers. The problem of multiplexing large amounts of data, exceeding the processing capabilities of most large-scale, real-time computers is discussed in detail. Implementation of hardware solutions to multiple Pulse Code Modulation (PCM) link multiplexing is dealt with. Use of firmware algorithms to reduce preprocessor front-end loading, as well as through-put reduction is discussed. The paper covers the different techniques used to take advantage of modern firmware preprocessor/multiplexers to select data for real-time computer processing.

INTRODUCTION

In designing a large telemetry system used for processing multiple PCM data streams, the problem of computer loading becomes a major factor. Bit rates for streams in the 2-3 Megabit/second range are routed to a computer in real-time with an aggregate word rate of 1-1.2 Megawords. This exceeds the computer's bus capability without considering system overhead. It becomes apparent that a front-end device is required to multiplex the streams and select only the data parameters required for real-time processing. In short, we multiplex several PCM streams into one, and screen out all undesired data to reduce the processing load on the host computer.

Word selection from each PCM stream is accomplished at the PCM decommutator level. The selected words are then multiplexed by the preprocessor and subjected to various algorithms before going to the host. The preprocessor passes the data parameters directly to the host's memory using an identification tag as the memory location. The host's software selects a parameter in the predefined memory location, logs it to disk, and processes it for graphic displays.

PREPROCESSOR DEVICES - THE MULTIPLEXING TECHNIQUE

It is now practical to use preprocessor devices for multiplexing high speed, multiple PCM streams from already decommutated data. Normal PCM decoms send their data directly from the decommutator into the computer via a computer interface. Where multiple streams are decommutated, the solution is to use multiple computer interfaces for processing the data (Figure 1). This technique basically funnels raw data from a decommutator device into the computer for sorting and tagging. This forces the host computer to do some type of subframe decommutation. This is usually a time-consuming procedure, with high computer system overhead and extensive programming required to support multiple streams. Once the stream count gets beyond two or three PCM streams, the interfaces consume a high amount of computer time and become costly in both hardware and software complexity. To reduce computer loading, the goal is to achieve the merging or multiplexing of multiple PCM streams, processed by different PCM decommutators, into a format a computer can easily use. When stream quantities reach four to six in magnitude, especially with high-rate PCM, the load on the host computer becomes so high that real time functions are limited. This limitation is especially true in the mini-computer class, where high overhead operating systems are normally used. An alternate solution to directly bring the output of multiple PCM decoms into a computer, is to install a preprocessor between the decommutator outputs and the computer.

The preprocessor takes multiple streams (many units are available today with up to six-stream merging capability) and merge these PCM streams into one, so that the host computer can process them through one interface in an expedious manner (Figure 2).

SUBFRAME PROCESSING

An important feature of this multiplexing technique is that most preprocessors on the market support multiplexing with subframe and frame information identified by the incoming decommutation. Basically, the subframe decommutation occurring in the PCM decommutator is utilized by the preprocessor using hardware techniques (Figure 3). This technique prevents the host computer from doing any unnecessary subframe decommutation. Words processed by the preprocessor are, therefore, identified by their frame and subframe before being multiplexed into the output stream created by the preprocessor unit. The preprocessor, therefore, can handle subframe and frame words selection in up to six high-speed PCM streams.

PREPROCESSOR FRONT END LOADING

The only factor limiting word rate is the front end of the preprocessor. The multiplexing process for each individual stream loaded into the preprocessor can be overloaded. These front ends usually are capable of handling four- to six-hundred kilowords, with burst rates up to 1.2 megawords. This limitation, however, can usually be overcome by doing word editing in the PCM decommutator before the data is transferred to the preprocessor. That is, only the words that need to be preprocessed before entering into the host computer are selected. Even when front end rates are exceeded by the multiple PCM streams, data is lost but synchronization is not. This is an important feature because software restarts for initialization phases are not required.

MULTIPLEXING USING ALGORITHMS

Another powerful tool in data multiplexing using preprocessors is the ability of many units to select data based on preprogrammed parameters (Figure 4). This selection is usually done in the preprocessor firmware by arithmetic means. Techniques for accomplishing data discrimination apply algorithms for limit testing, bit pattern matching, and other various techniques to ensure that data is not multiplexed into the output channel unless it meets the preprogrammed criteria. This discrimination of data by the preprocessor has the ripple effect of slowing down the output rate to the host computer. This effect can be very powerful because only desired data is multiplexed and sent to the host. For example, a limit is placed on a data parameter, so that the host processor is only notified when the parameter is an out-of-limit condition. This pseudo-intelligence by the preprocessing device ensures the host computer's freedom for performing routing and more sophisticated processing on the incoming data stream.

DATA IDENTIFICATION

Another powerful feature of multiplexing, multiple PCM streams within a preprocessing device is the ability of the device to identify the data to the host (Figure 5). This identification is done by using a preprogrammed tag along with the data fed to the host computer. This tag or identifier can be used as an offset address to a computer-generated address and can therefore be used as a direct-memory access tool. This processing allows data to be entered in memory directly from the multiplexing device without special software processing. This tool, often called "externally specified address," minimizes the overhead in the host computer related to routing the data from the preprocessor/multiplexer into specific locations in computer memory. Another method, very similar in nature, is to identify the tag and the data in a sequence with the tag following the data, then the next data point, the next tag etc. This technique allows the computer's host software to identify each data point by finding the tag in the buffer being loaded by the interface channel. This mode is often used in block transfers as well as sequential transfers on a word-by-word basis. Blocks of data and tag are loaded into the host computer from the multiplexer/ preprocessor, and when this block is completed, a signal is sent to the host to indicate that a group of data are ready for processing. Again, the use of the multiplexer to identify the data by placing a tag with it is an extremely powerful feature. It reduces software complexity for routing, allows ease of real-time program operation, and minimizes computer overhead. A third technique routes specific parameters into arrays before transferring them to the host computer. These arrays contain continuous samples of the same data parameter so that one array basically contains a particular parameter within a specific PCM stream. These parameters are then down-loaded to the host device in a group, so that they can be examined for averaging and multiple-sample algorithm processing. This technique is used often with array type processors as the host. To summarize, the output of the multiplexer/preprocessor has basically three different forms--all of them essentially identifying the data to simplify the tasks within the host or processing computer.

CONCLUSION

Software considerations, when using preprocessors as multiplexing devices, are, of course, elaborate in nature. Usually, a compiler is required to support the preprocessing device and to identify the specific parameters and their algorithms for multiplexing. Since this is a pretest function, it has no restriction on the real-time processing load by the preprocessor or the host. By using the multiplexing technique, the preprocessing device will simplify the real-time software events within the computer. The solution is to offload host processors by using preprocessors to multiplex the PCM streams required in today’s telemetry environment. The present trend in bit rates and therefore word rates is steadily increasing in frequency. More and more data needs to be sampled at a faster and faster rate. The trend is to increase the bit rates of the serial streams into the megabytes-per-second range. This increase in word rates requires use of preprocessors to multiplex and select the data for processing by the host computer. Another factor that contributes to the need for these devices is the use of the host for complicated real-time graphics displays. The use of the preprocessor to select the data for multiplexing is probably one of the most powerful forms of multiplexing used in today’s telemetry processing.

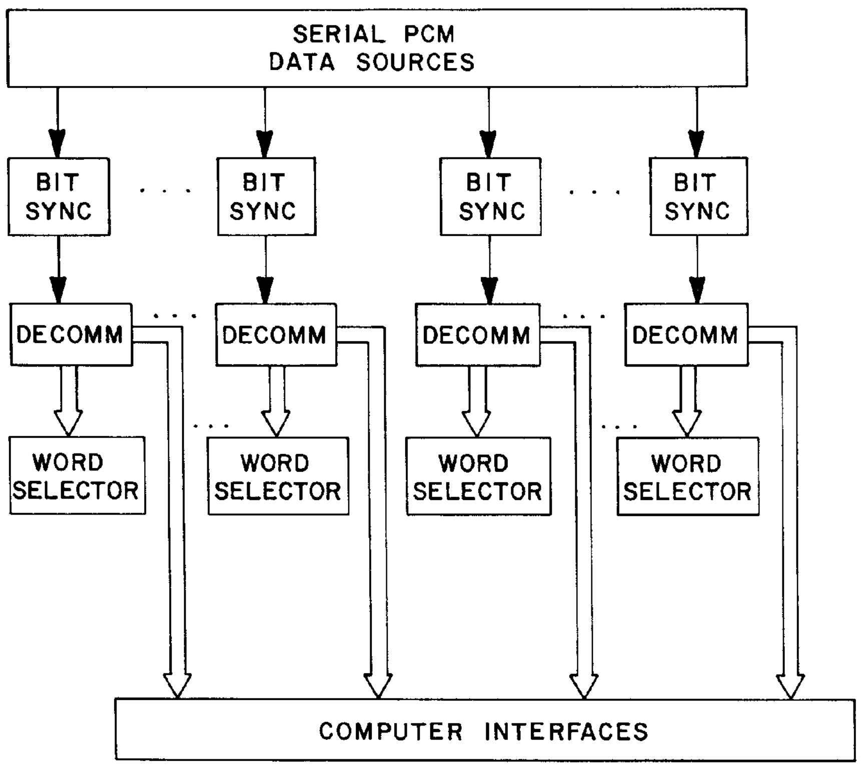
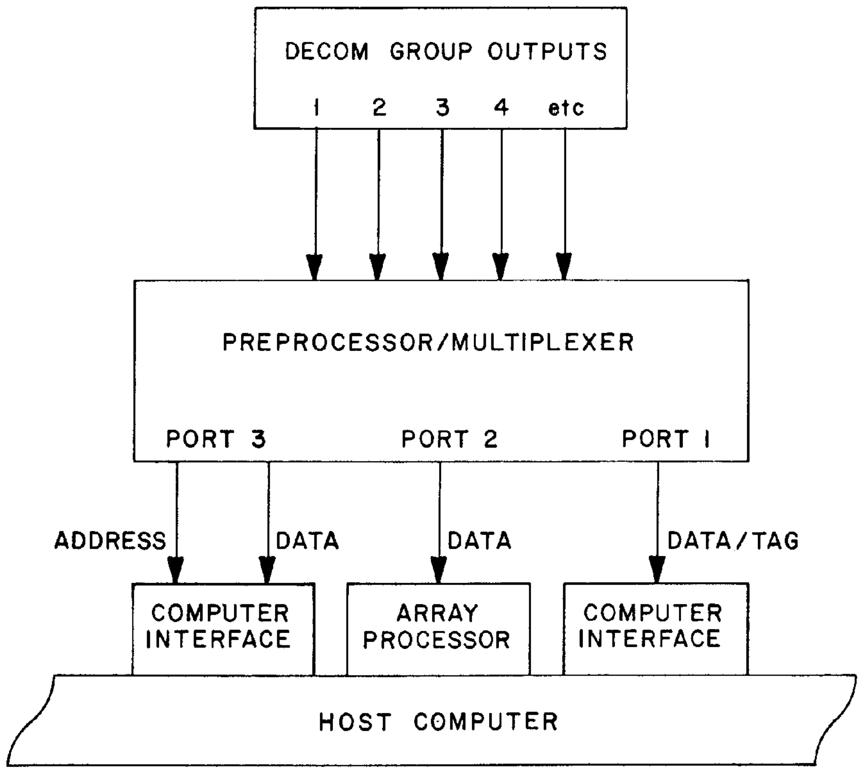
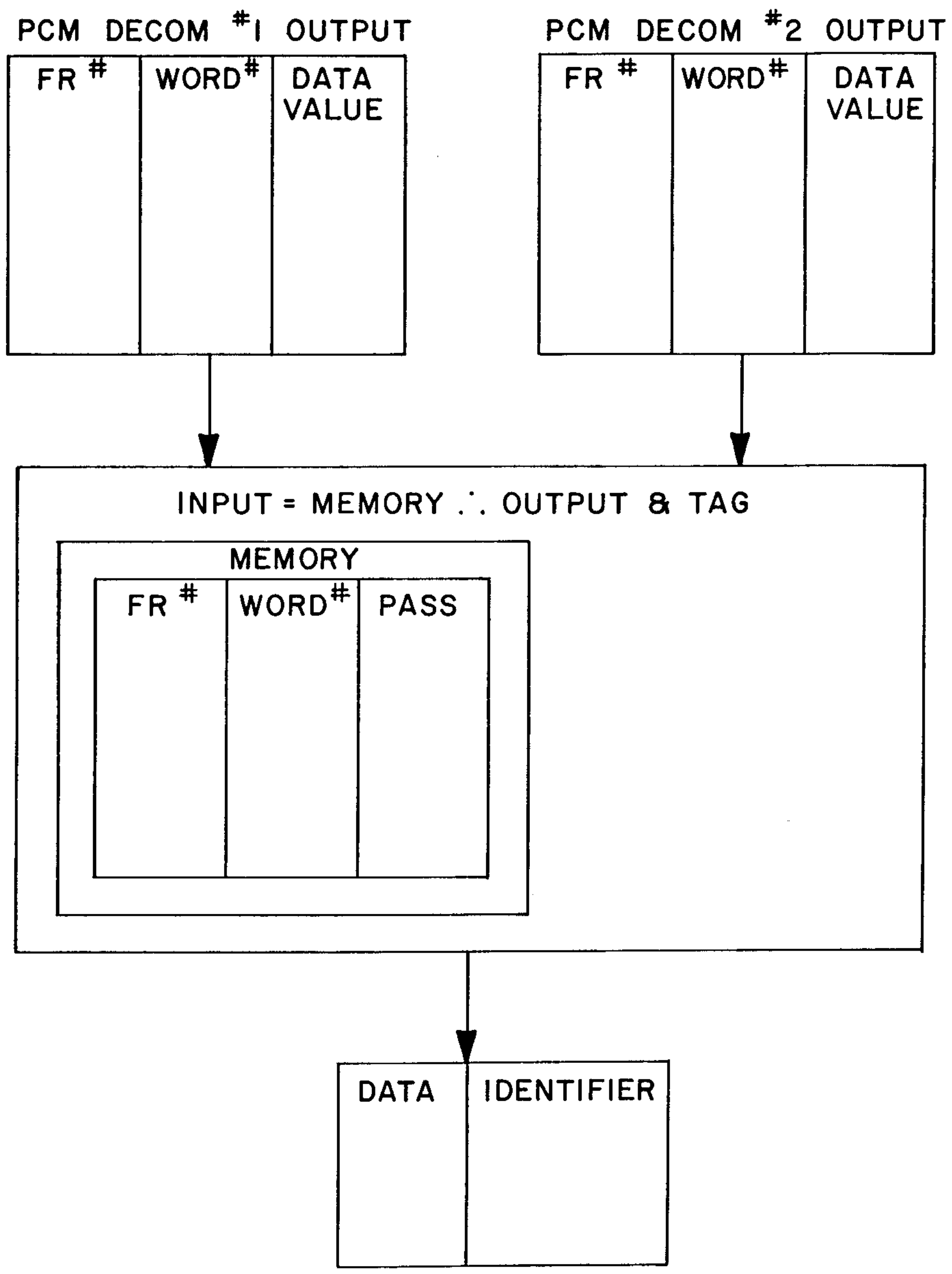


Figure 1



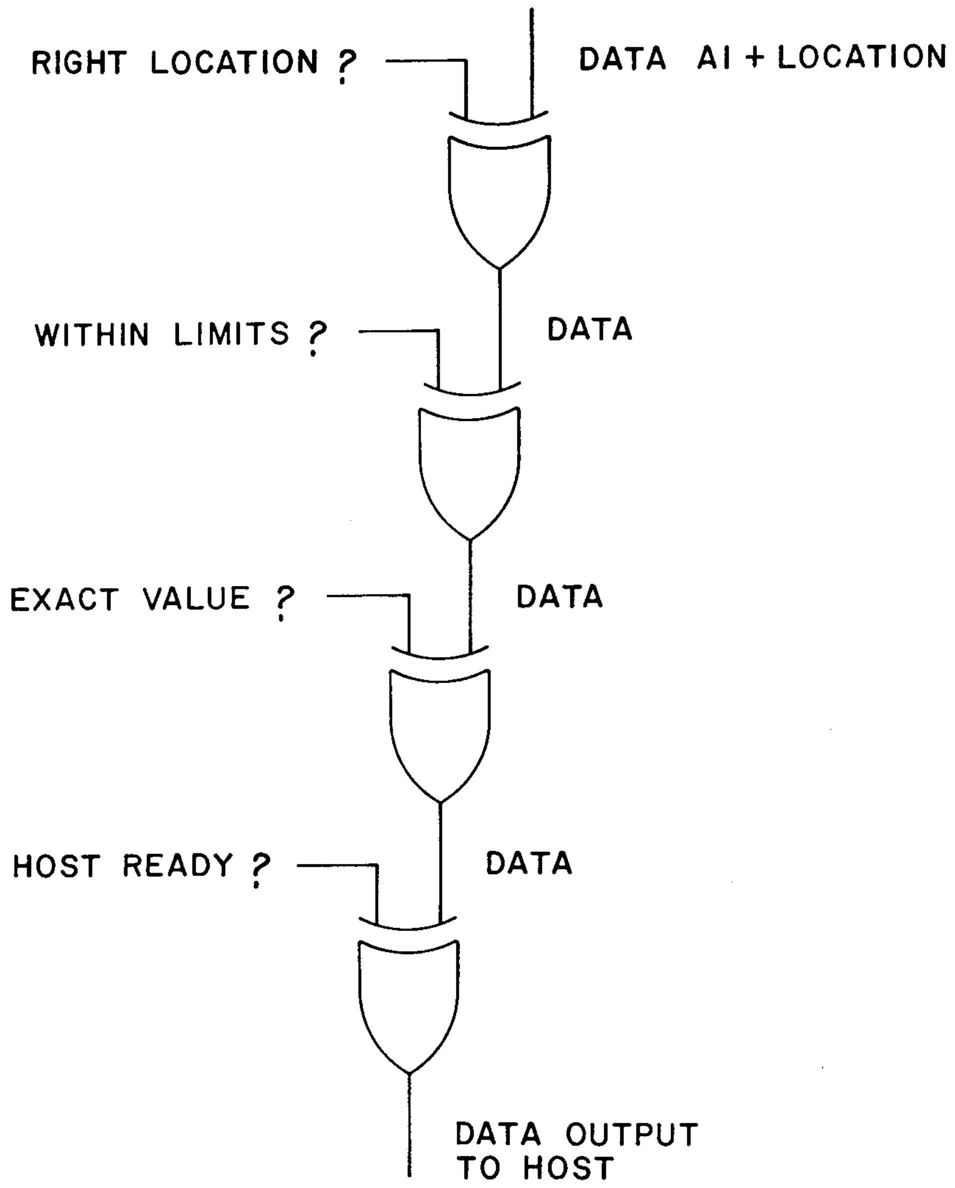
MULTIPLE PCM STREAM MULTIPLEXING

Figure 2



MULTIPLEXING BY LOCATION

Figure 3



PSEUDO INTELLIGENT MULTIPLEXING

Figure 4

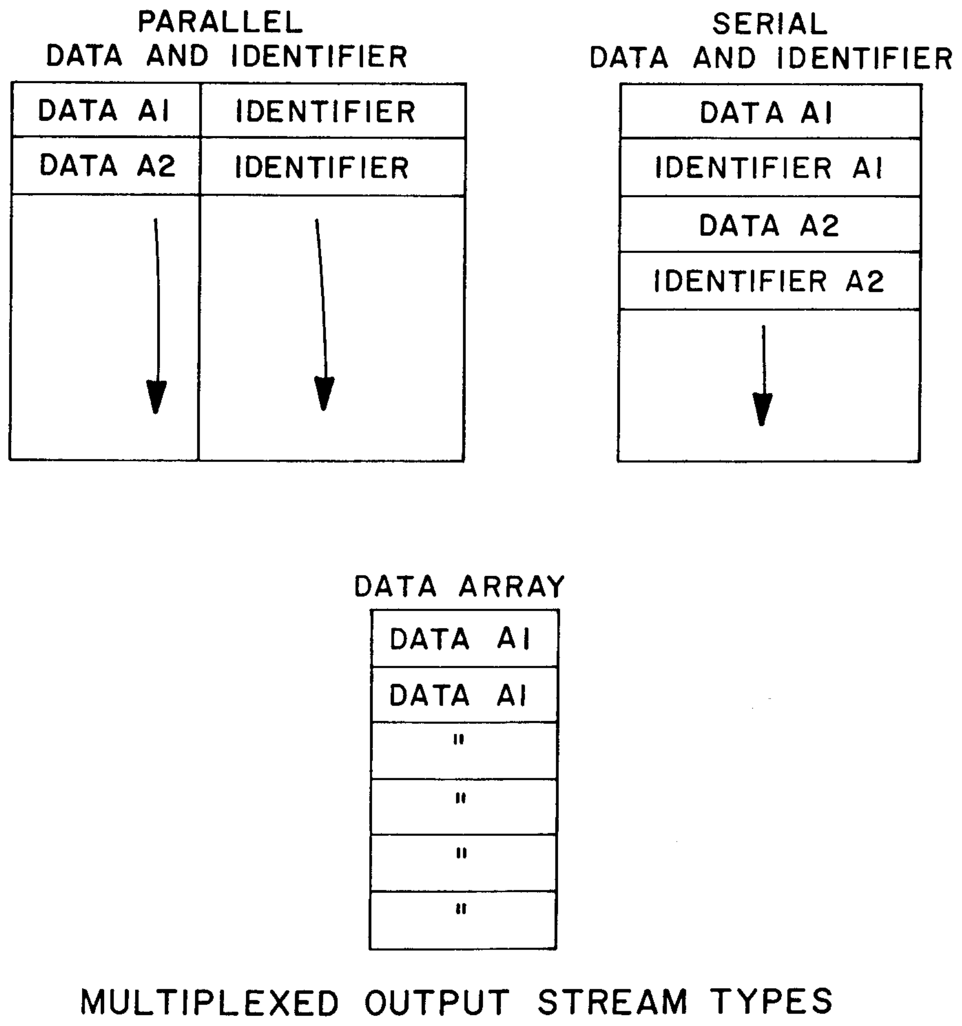

PARALLEL
DATA AND IDENTIFIER
DATA A1
IDENTIFIER
DATA A2
IDENTIFIER
SERIAL
DATA AND IDENTIFIER
DATA A1
IDENTIFIER A1
DATA A2
IDENTIFIER A2
DATA ARRAY
DATA A1
DATA A1
"
"
"
"
MULTIPLEXED OUTPUT STREAM TYPES

Figure 5