OFF-LOADING THE HOST COMPUTER THROUGH FLEXIBLE DMA INTERFACE

STEPHEN J. NICOLO
PRINCIPAL ENGINEER

AYDIN MONITOR SYSTEMS
502 Office Center Drive
Fort Washington, PA 19034
U.S.A.

ABSTRACT

As data rates and system throughput requirements continue to increase, more and more attention must be given to ways of off-loading the host computer by shifting tasks to the front-end preprocessing subsystem. In addition to some of the more common tasks like data compression and EU conversion already performed in the front end, there is the time consuming task of organizing telemetry data. Once relieved from this secondary task the host can solely attend to its primary task of application processing. This paper describes an intelligent DMA interface (CPI007) which permits the automatic building of various types of array buffers in the host computer.

This flexible high-speed device uses an EPROM based, bit slice microengine utilizing parameters stored in its operational store RAM during setup to build the array buffers. The interface is implemented on a single module in the front-end preprocessing subsystem and was developed for those mainframe computers that can be configured to accept address/data inputs for DMA to system memory (e.g. Gould Sel, DEC).

With this type of architecture, algorithms may easily be written to accommodate a wide variety of data organization and transfer requirements. Along with the technical description of this device, actual data array buffering problems and solutions will also be addressed in this paper.

INTRODUCTION

Since the early 1960s Aydin Monitor Systems (AMS) has utilized a Data Flow Architecture in its front-end telemetry preprocessing subsystems. The latest and most powerful implementation of this architecture is realized in the AMS Series 2000 (S2000) with its high-speed (6 million transfers/second) Mercury Bus. In this distributed processing subsystem a modular approach is taken with each element on the Mercury Bus performing its own particular function such as decommutation, data compression, EU conversion, limit checking, or output. With this type of architecture each measurement is tagged and merged onto the bus in a broadcast fashion. The tag then directs the data value to each particular process. Once a process is performed, the data is re-tagged and broadcast back onto the bus directing it to another module for further processing or output.

In the past, output modules would pass both raw and EU-converted data directly to the host, or sequentially DMA these values into host memory. The time consuming task of sorting and directing this data was then left up to the mainframe computer. This wasted computer time and greatly reduced the overall processing power of the system. With this in mind, and with the onset of very powerful and compact bit slice processors, VLSI devices, and CMOS memories, AMS developed the CPI007 intelligent array builder interface on a single S2000 module.

The CPI007 Array Builder Interface is the evolved culmination of computer interfaces developed by AMS to transfer information from the S2000 multiprocessing subsystem to large-scale computers. It is an intelligent DMA device with a bit slice architecture where the processor is controlled by algorithms stored in microcode PROMs. In this way the board is very versatile, because in addition to the algorithms presently provided, additional algorithms can be provided to meet any specific customer’s needs.

S2000 MERCURY BUS

The Mercury Bus with its Data Flow Architecture has 16 bits of address/ID, 32 bits of data, and 9 control signals. Two types of transfers can occur on this bus: setup and broadcast. Setup transfers are used to initialize and configure each module. Broadcast transfers are used to pass tagged data from module to module for processing or output. Tagged broadcast data transfers are output onto the bus by Mercury Bus masters such as frame synchronizers, analog-to-digital converters, time code translators, and interface control modules. Mercury Bus slave devices such as digital-to-analog converters, digital display generators, and computer interface modules capture data based on the ID tag. Modules that perform processes on the data and then rebroadcast it back onto the bus, such as digital signal processors, floating point processors, and data compressors are both masters and slaves. Each slave device has a word selector memory on it (64K X 1). During setup the RAM adderess location corresponding to the ID tag which is to be captured is set. During broadcast transfers the ID tag is used as an address to a word selector RAM location. If that particular memory location was set by the user, the tag, along with its corresponding data value, will be captured for processing or output.

CPI007 COMPUTER INTERFACE FUNCTIONAL DESCRIPTION

The CPI007 Array Builder Interface plugs into the S2000 Mercury Bus interfacing the S2000 preprocessing subsystem chassis to the larger host computer. The module was developed for the Gould Sel 9131 HSD port, but can easily be adapted to other mainframe computers that similarly support direct memory addressing from external devices. The module can build and keep track of up to 2048 arrays, buffers or current value tables (CVTs) in the host computer's memory, supplying a calculated address along with each measurement. It also has the capability to map any of the 64K ID tags to any one of the 2048 arrays or buffers. This gives the user total flexibility over ID tag selection.

This module also has a time word input. It accepts 48 bits of serial time data at 10-microsecond intervals from a time code translator. This time data can then be accessed by the ALU and stored in host memory along with the measurement. Time data is accessed by the ALU in two words: the fine time word (16 LSBs of time word), and major time word (32 MSBs of time word).

The CPI007 is a fully bidirectional device. It can be a setup or broadcast master on the Mercury Bus under full control of the host computer, or a broadcast slave, capturing data from the bus and transmitting it to the host. Upon powerup the output format is selected, the word selector memory is set up, and all parameters required to build the specific arrays or buffers are loaded into the operational store RAMs using setup transfers. Since the CPI007 is a fully bidirectional device, these setup transfers could come from the host computer or the Mercury Bus. Along with the word selector memory, a 64K X 11-bit operational store map address RAM is also loaded. This map address pointed to by the ID tag during broadcast transfers is loaded into the FIFO along with the ID tag and data. It is then a pointer to one of 2048 sets of parameters, stored in operational store RAM, required to build and keep track of that particular array or buffer.

On the output side of the FIFOs is the bit slice processor circuitry. The processor is a 32-bit device with an ALU, and microsequencer, executing microprogram instructions stored in its microcode PROMs. All output buses are routed through the ALU to an output register. The ALU has access to the operational store RAMs, the time word, the captured ID and data words, a 16-bit Mercury Bus read/write register, and a 24-bit base address register written into by the host during command transfers. In this way the processor can perform arithmetic logic functions on not only the DMA addresses and the operational store RAM parameters, but also on the data if required.

A variable length clock generator controls the microinstruction cycle length. Each instruction's cycle length is selected by 3 bits of the microcode. The majority of the instructions can be executed in 100 nanoseconds (10MHz) with the minimum and maximum being 75 and 175 nanoseconds respectively. Depending on the algorithm, the processor can run at an average of 7 to 10 MIPS.

STANDARD ALGORITHMS

The microinstructions for the algorithms run by the processor are stored in the microcode PROMs. This makes the module easily adaptable to specific array buffer building situations. Standard algorithms provided with this module support sequential mode, straight external mode, current value table building, single parameter array building, and block parameter array building. These algorithms will support most computer functions required by the host, such as data logging, array processor buffer building, current value tables, and data buffers for inter-computer transfers. Each of these algorithms are explained in greater detail in the following paragraphs. Array buffers built by the single and block parameter array algorithms are illustrated in figures 1 and 2.

-Sequential Mode Algorithm

The sequential mode algorithm is used for data logging or archiving. This is the only non-DMA mode. In this algorithm the CPI007 processor does no host memory address calculations. The CPI007 passes each captured ID tag and data value directly to the host in two 32-bit transfers. The first transfer consists of the fine time (16 LSBs of the 48-bit time word), and the 16-bit captured ID tag. The second transfer consists of the corresponding 32-bit data value. When enabled, major time word transfers can also be merged into the data stream on the occurrence of an external trigger pulse.

-Straight External Mode Algorithm

The straight external mode algorithm is basically a CVT-type algorithm. This is a DMA mode with host memory addresses provided by the CPI007. In this mode the parameters in the operational store RAM are not used. A base address is loaded into a register (base address register) in the CPI007 by the host. The captured ID tag is then converted to an offset and added to this base address to get the physical memory location. This physical address is then sent to the host along with the measurement in two 32-bit transfers.

This algorithm makes the module upword compatible to previous AMS computer interface modules. The algorithm could also be used in systems with large numbers of CVT values which exceed the 2048 that can be stored in operational store memory. This approach requires a ridgid ordering of data within the host buffer, with any changes in this order requiring the user to reprogram the ID tags in the subsystem. Reprogramming of ID tags can be time consuming and may affect many of the processing modules in the subsystem.

-Current Value Table (CVT) Mode Algorithm

The CVT algorithm is mainly used by the host computer for display purposes. In this algorithm the Mercury Bus ID tags are used as pointers to one of 2048 physical host memory addresses stored in the operational store RAM. The ALU is then mapped to one of these physical addresses. Because the Mercury Bus ID tag is just a pointer, physical host addresses for CVT measurements may be changed very easily by changing them in the CPI007's operational store memory without having to reconfigure bus ID tags. If time tagging is selected, the CPI007 will output a 32-bit major time word to the location in host memory immediately following the measurement.

-Single Parameter Array (SPA) Mode Algorithm

The SPA algorithm is basically a CVT with depth. This buffer is used to keep a history of EU-converted measurements in the memory of the host for display. The Mercury Bus ID tags are pointers to 2048 sets of operational store memory parameters used to build the array buffer. During module setup for each array, the user must program the physical host computer start address and word count (depth of array). The SPA algorithm then generates and maintains the correct memory address where that particular sample is to be stored. When the array buffer length is reached, the SPA algorithm will begin overwriting the samples at the beginning of the buffer area. The CPI007 will also provide a current value pointer for each of the array buffers that have been defined. The current value pointer resides at the first location of the array buffer, and indicates the location within that buffer of the last parameter sample transferred. The current value pointer for each buffer is updated with each sample transferred to that buffer. If time words are enabled, a 32-bit major time word will also follow each array sample. The array buffer for each measurement will then consist of a data sample followed by a time word.

-Block Parameter Array (BPA) Mode Algorithm

The block parameter array algorithm will build a buffer made up of a group of defined measurments that are repeated in the array. This algorithm was used by one customer, for example, to recreate in host memory each input stream with its EU-converted measurements. The BPA algorithm permits up to 64K unique measurements to be defined as a group, and up to 64K repetitions of this group to be defined as a block. Up to 2047 block buffers can be built in host memory. The captured Mercury Bus ID tag is a pointer to one of 2047 sets of parameters stored in operational store memory used to keep track of the buffers. The algorithm can be programmed with a start ID tag that will determine the first parameter within each group to be transferred to the host. This will enable the measurements within the group to be synchronized. If time words are enabled, this algorithm will place a time word at the last location of each group upon completion of a group. Along with the array of measurements, an historical array of groups completed will also be built. This array will contain the history of the groups completed. Upon completion of a group, the group number and block identifier will be written to the historical array. The historical array is also supported by a current value pointer of the array. This pointer indicates the last updated location in the historical array. This allows the user to easily determine which groups in which blocks are complete, so that they may be processed or possibly passed to a special graphics subsystem. The CPI007 can also be programmed to generate an interrupt after N numbers of groups are complete.

Non-standard Algorithms

The flexibility of this module is best realized in situations where the specific application does not fall into the category of any of the standard algorithms. In these special cases non- standard algorithms are written to provide a solution to each unique application problem. In the following paragraphs one specific application problem and its solution through microcode change is described.

-Non-standard Application Problem

One example of a microcode solution to a non-standard application problem was formulated for a system having multiple data streams being played back from tape. Along with the data on each of these tapes, IRIG time was also recorded. Each stream was fed into a frame synchronizer and a time code translator. A frame mark signal from each frame synchronizer would merge a time word with a unique ID tag onto the Mercury Bus from the time code translator. The problem was that the external hardware could only synchronize these tapes to within one second of each other. The user wanted AMS to build several ring buffers in the host memory that would store up to a second's worth of data. AMS was also required to devise a strategy for their software to easily correlate these buffers with time.

-Non-standard Application Solution

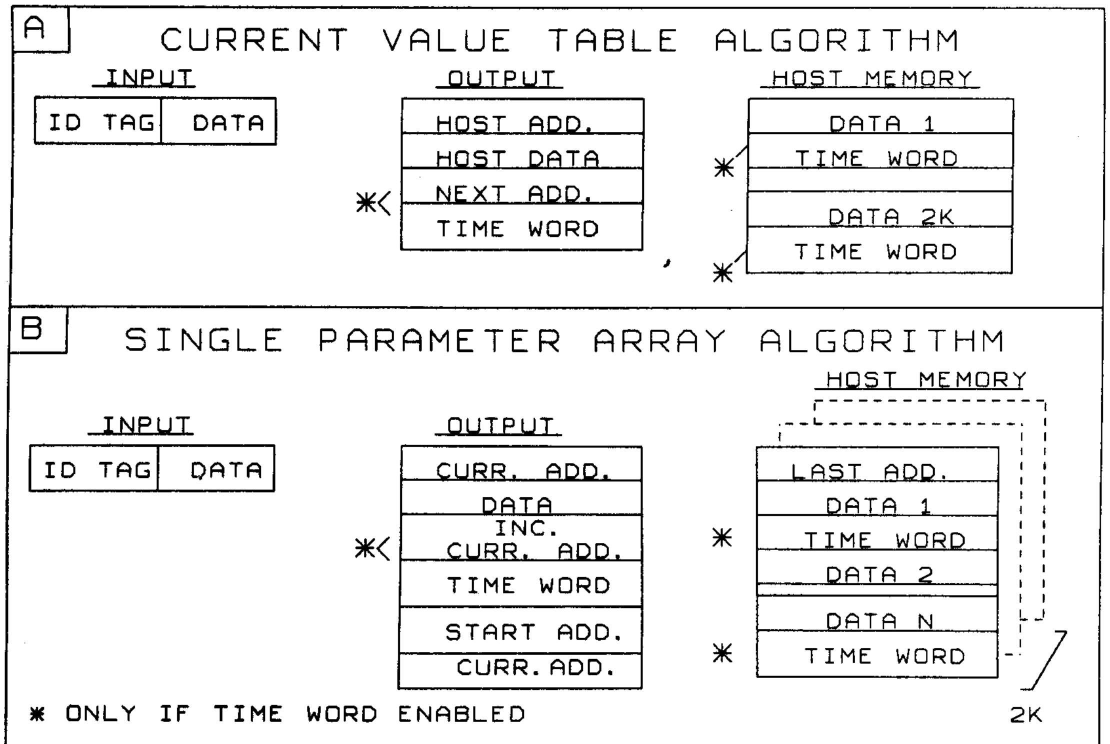
The solution to this problem was an algorithm similar to the Block Parameter Array algorithm. In this algorithm each stream of data would be stored in its own ring buffer in host memory. Each frame of data would be identified as a group. Group boundraies would be distinguished by the CPI007 using the unique time word ID tag, output by the time code translator on frame mark, for that particular stream. Along with these measurement buffers an historical array of groups complete would also be built. This array would contain the history of groups complete, along with the time word associated with each unique ID tag. The CPI007 could also be programmed to interrupt after receipt of N number of unique ID tags. The host could inspect the history array buffers stored time words, along with the completed group identifiers, and correlate each group with time. The host software could then process groups of data in chronological order.

CONCLUSION

The CPI007 Intelligent Array Builder Interface is a flexible and powerful tool that provides standard algorithms that meet most system application needs. Because of its EPROM-based, bit slice, microinstruction-driven architecture, this module can be easily adapted to solve any specific host computer data organization application problem. For many of the newer, real time, high throughput systems, the savings in processing time far surpasses the implememtation cost of the device.

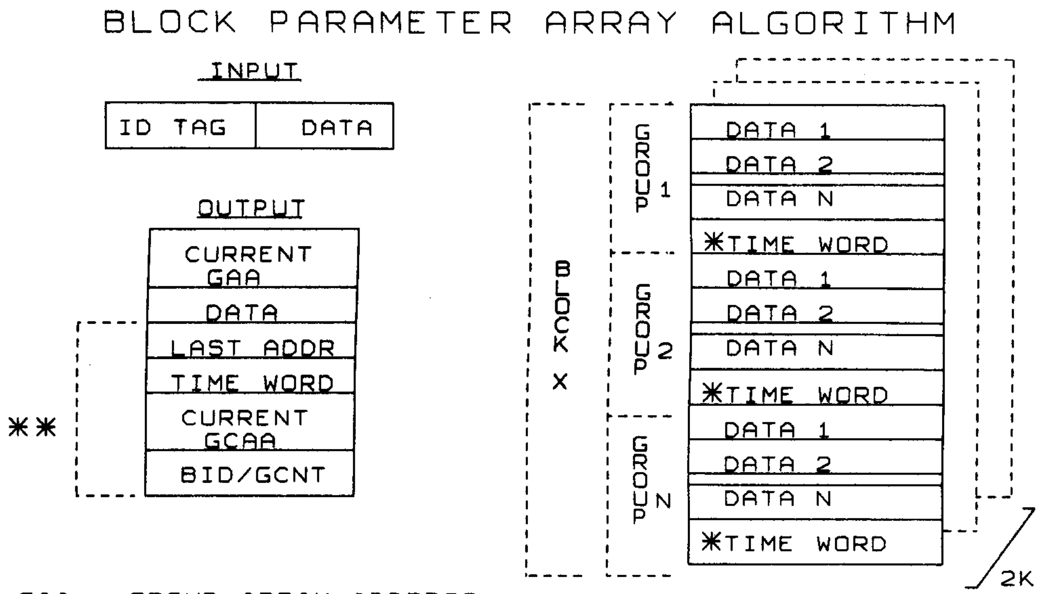
ACKNOWLEDGEMENTS

The author would like to thank Edwin Snyder, Director of Data Aquisition Sales at Aydin Monitor Systems, Chris Dehmelt, Software Engineer at Aydin Monitor Systems, and Ben Weinberg, of Computer Science Corporation (initial customer's Technical Representative), for their support over the course of this project.



✱ ONLY IF TIME WORD ENABLED

FIGURE 1



GAA - GROUP ARRAY ADDRESS
GCAA - GROUP COMPLETE ARRAY ADDRESS
BID/GCNT - BLOCK IDENTIFIER/GROUP COUNT
✱ - ONLY IF TIME WORDS ARE ENABLED
✱✱ - ONLY UPON COMPLETION OF A GROUP

FIGURE 2