Data Logging Solution for Digital Signal Processors

Brian Newberry
Nekton Research, Inc.
BNewberry@NektonResearch.com

James M. Conrad
University of North Carolina at Charlotte
jmconrad@uncc.edu

Abstract

Digital Signal Processors (DSPs) are some of the most powerful processors in the world. With clock speeds topping 1 GHz and architectures capable of performing multiple instructions per clock cycle, they are ideally suited for working with audio and video signals with high data rates. The signal processing algorithms that are used to analyze such data are complex and can be difficult to debug if results are not as expected. Having the ability to store the data that is being analyzed by the DSP allows for post processing of data so that the algorithm can be perfected outside of actual testing conditions. This capability of "playback" can drastically reduce the time and cost of system testing. For this purpose, an embedded DSP data logger was designed to store audio data that is being collected and analyzed by a DSP. The data logger is designed to download audio data from the DSP at a high rate, store the audio data as a binary file on a static storage medium, and be able to upload the data to a separate PC from the static storage medium.

1. Introduction

The Global Positioning System (GPS) has solved many of the problems of navigation that have faced mankind for thousands of years. The ability to find the exact location of any device on the planet has also changed the way that autonomous mobile systems operate. However, GPS systems have their limitations. The signals sent from the GPS satellites cannot penetrate the earth or into the water that covers nearly two thirds of the earth's surface. This means that for underwater vehicles, the problem of navigation is not yet solved. Manned submarines use gyros and inertial navigation systems that are able to provide navigation information, but these systems are prohibitively expensive and large for use on small autonomous underwater vehicles (AUVs). One possible solution to the problem of underwater navigation is to use buoys, placed in known locations that emit specific frequencies into the water. By finding the Doppler shift that an AUV perceives relative to that specific carrier frequency emitted by a source and with the added knowledge of the velocity of the vehicle, the heading relative to the frequency source can be determined. Using multiple buoys each with their own specific frequency would allow the exact location of the AUV to be determined. This concept is currently in the development phase and much testing remains to be done. The algorithm that is being used for finding the Doppler shift and then computing the heading relative to the frequency source is complex and the audio signal that is received by the system may require conditioning because multi-path reflections and background noise may be significant in certain underwater environments.

The purpose of this project was to provide a means of data logging for the system that will be analyzing the frequency data collected by the AUV. The ability to log the analog data gathered by the system would allow for post processing using actual test data rather than simulated data. For this particular application, this will represent a significant reduction in the cost of testing. A complete system test would require at least two people an entire day to perform and would necessitate a great deal of setup and travel. If enough data could be collected and recorded in one day of testing, then that data could be archived and used to perform tests on algorithm refinements in a laboratory setting while not sacrificing any of the characteristics of the actual testing environment. The data logging part of the system could eventually be removed completely after the algorithm is perfected and no recording of audio data is desired.

Since this system is being designed for an embedded application, there are certain restrictions on the design and the components that can be used. The entire system must be as low power as possible and be able to fit inside a five-inch diameter cylinder. The low power constraint comes from the fact that all of the power the system uses will have to be carried on batteries inside of the AUV. The five-inch size constraint comes from the size of the particular AUV that the system is going to be tested. These two constraints limit the choice of processors that can be used and also what sort of storage medium can be used to log the analog data that is being collected. It is also important to have an easy method of extracting the logged data because the entire design will be enclosed in a watertight compartment that will not be easy to disassemble. The transfer of data out of the system to a separate PC for post processing must be done using a high bit rate because of the large amount of data that will be accumulated while the system is in operation.

The data collection and analysis need to take place at very high speeds and thus will need to be run on a very fast processor. A Digital Signal Processor (DSP) is a good fit for this system because of its high performance low power consumption compared to conventional processors. A Pentium III processor running at 1 GHz will consume 29 Watts of power while a 6416T Texas Instruments (TI) DSP running at 1 GHz with all supporting hardware installed will consume about 2 Watts [1,2]. Conservatively, a Pentium based system would drain the available power supply fifteen times faster than the same system using a DSP.

Although using a DSP makes perfect sense for this system when considering power constraints, this choice introduces several drawbacks. Whereas there is wide operating system support for general-purpose processors like the Pentium, there is little to no operating system support for DSPs. This limitation is important to this system because an operating system simplifies the task of storing data onto a static storage medium. Operating systems hide the complexity of file writes to static drives by using their own drivers to interface software and hardware whereas to use the DSP to directly store the analog data that is being created, new drivers must be written specific to the particular DSP and the static storage medium being used. This is a nontrivial task but several solutions have been commercialized.

Another option is to port the data off of the DSP onto a separate system. This option frees the DSP program from the implementation of file system drivers. This introduces another problem though in the transport of the data from one system to another. The transfer rate must be very fast and ideally use a standard native to the hardware of the data logging system and the DSP. The Serial Peripheral Interface (SPI) transmission protocol was chosen to port the data off of the DSP and onto a separate single board computer (SBC). This solution was chosen because it meets all requirements and leads to low power and small size.

2. **Audio Data - System Requirements**

The audio data gathered from the underwater environment is collected using a hydrophone. The hydrophone functions exactly like a microphone except that it works underwater, converting sound waves propagated through the water into an analog electrical signal. This analog electrical signal needs to be converted into digital data so that the DSP can analyze it and find the Doppler shift from the carrier frequency. It was determined before the design phase of this system that the signal would need to be sampled at 100 kHz using a resolution of 16 bits to achieve the desired frequency resolution. This equates to a raw data rate of 200 KB per second and the DSP needs to have enough RAM to store an entire second of sampling. Also, it is desired that whatever static storage medium is chosen to hold the logged data be large enough to hold five minutes of sampled data (60 MB).

3. System Specifications and Design Choices

The DSP platform chosen for this project was the DSK6416T, a starter kit from TI based around the 6416T DSP. The 6416T is a 1 GHz fixed point DSP with 1 MB of onboard RAM. The 6416T also has three multi-channel buffered serial ports (McBSPs) that can implement the SPI interface. The development kit gives the 6416T access to 16 MB of offboard RAM, 512 kB of flash memory for storing program information, and is programmed using C over a JTAG interface. The programs written for the DSP are easily downloaded onto the flash memory on the board via a USB connection from a PC that serves as the JTAG connection to the system. The entire DSK is only four and half inches wide and so will fit into the five-inch cylinder.

The analog to digital converter (ADC) chosen for the system is an audio codec made by TI, the TLV320AIC23B. This codec is an ADC specifically designed to work with the frequency range of the carrier signal for this system making it an ideal choice. It operates at a 16-bit resolution and can sample as fast as 96 kHz. A TLV320AIC23B is conveniently included onboard the DSK6416T and so no additional hardware costs are incurred by using it. Figure 1 below shows the DSK6416T with the audio codec circled.

Compact flash was chosen as the storage medium for the logged data. Compact flash is a very rugged storage solution due to its static design, using no moving parts, and can be purchased with the capacity needed to store five minutes of data relatively inexpensively. The manufacturer of the compact flash used for this project, Silicon Systems, claims that the commercial grade compact flash card can withstand a 1000 G shock and up to 15 G constant vibrations without affecting its performance. The compact flash card uses only 0.12 mA during standby operation and less than 35 mA when it is being written-to or read-from. It has read and write speeds of 6 MB/second and 8 MB/second, respectively. This more than meets all the requirements needed for the system with the added bonus of superb ruggedness and low power. A 256 MB card was installed in the system to give 22 minutes of storage capacity allowing the system longer testing times between data dumps.

There are two basic options for controlling the compact flash card: use the DSP directly or use a separate system. Using the DSP directly would necessitate the creation of file system drivers and the knowledge of the inner workings of the compact flash card. A commercially available solution was found that would eliminate this problem. A daughter card that will plug directly into the DSK6416T and hold the compact flash card is available off shelf, manufactured by a company named Applied Signal Processing. This daughter card cost $300 and was well within the budget of the project but the file system drivers that would allow the DSK to use the daughter card cost $12,000 for a single license [3]. The cost of this solution was deemed too high. Designing the system using a separate computer to store the data onto the compact flash card is much more economical but it introduces another design challenge: How does the data get transferred off of the DSP and onto the separate system that will store it onto the compact flash card? One solution is to use an Ethernet daughter card that was available for the DSK6416T. This, however, would have tripled the current hardware cost of the system, added complexity to the DSP code while stealing processing power away from the signal processing functions it would be performing, and used more power supporting the Ethernet hardware [4]. Alternatively, it was decided to use one of the McBSPs onboard the 6416T to implement a SPI port to stream the data to another system. This approach would require no additional hardware to be connected to the DSP other than the separate system itself, allow for data rates greater than 1 Mbit/second, and consume only 1 mW of additional power [2].

The storage medium controller using the current design needed to have an available SPI port, the ability to use compact flash as a drive, and meet all physical standards for the system. The TS-7200, a single board ARM based Linux machine was chosen to act as the compact flash controller. This system made by Technologic Systems is based on the PC104 standard measuring only 3.75 inches wide, uses a maximum of 2.5 Watts to operate, and has an SPI port, onboard Ethernet controller, compact flash holder, 16 MB of flash memory, and 32 MB of RAM. All these specifications make the TS-7200 an ideal solution for this system.

The operating system onboard the TS-7200 is a Linux 2.4 kernel, using only 5 MB of the 16 MB of flash memory that is available. It uses a COM port to act as a console for the system making it easy to interface with. The operating system also includes an FTP (File Transfer Protocol) Apache server that runs in the background. This can be used to download the data from the compact flash once it has been recorded. The FTP server allows anyone running a web browser to log into the \root directory on the system and extract the data. In tests, the TS-7200 was able to download 57.7 MB of data, the amount gathered in 5 minutes of testing, in approximately 68 seconds. This equates to a data transfer rate of 0.85 MB/second. This high transfer rate will decrease the downtime during testing to download data that has been recorded. It will be simple to route Ethernet access for the TS-7200 to the outside of the waterproof enclosure the system will be housed in allowing data downloads to be performed with no disassembly required.

The system when finished cost around seven hundred and fifty dollars. The DSK6416T represents 67% of the cost of the system. It is also important to note that when the concept has been thoroughly tested and the algorithm is perfected, the cost will be cut by almost 30% by removing the TS-7200 from the system.

Figure 1 shows the finished system. The board on the far left of the picture is the TS-7200. There are two small boards shown in the bottom right hand of the picture. One of these boards is a simple power supply built to power the system during testing. The other is a daughter card that holds a MAX232 Universal Asynchronous Receiver Transmitter (UART) to RS232 converter. This chip provides the DSP board a way to communicate with the outside world via a serial port.

Figure 2 below shows a block diagram of the system as it would be setup for testing including the power system and all external interfaces.

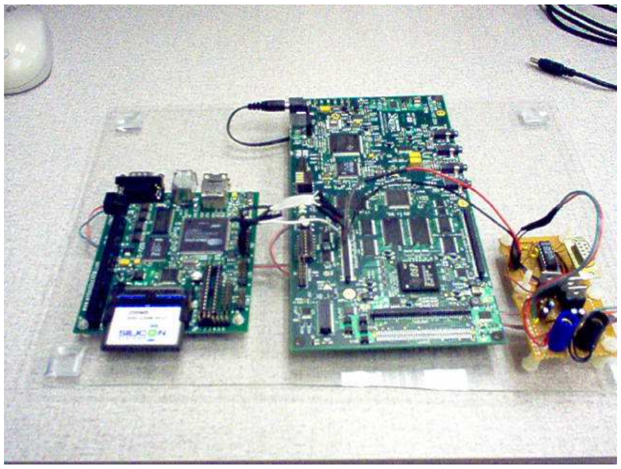
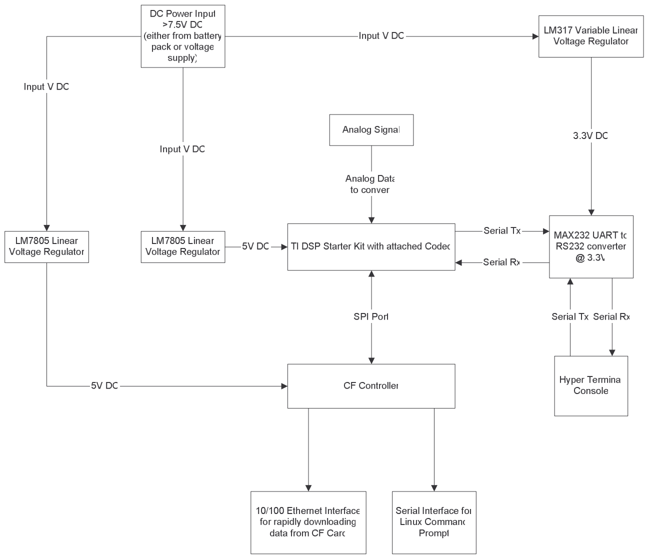


Fig. 1. Finished system set up for testing


DC Power Input >7.5V DC (either from battery pack or voltage supply)
Input V DC
LM317 Variable Linear Voltage Regulator
Input V DC
Input V DC
Analog Signal
3.3V DC
Analog Data to conver
LM7805 Linear Voltage Regulator
LM7805 Linear Voltage Regulator
5V DC
TI DSP Starter Kit with attached Codec
Serial Tx
Serial Rx
MAX232 UART to RS232 converter @ 3.3V
SPI Port
Serial Tx
Serial Rx
5V DC
CF Controller
Hyper Terminal Console
10/100 Ethernet Interface for rapidly downloading data from CF Card
Serial Interface for Linux Command Prompt
Fig. 2. Block diagram of system

4. Code Design

The system was designed with a very simple, single threaded code structure on both the DSP and the TS-7200 board. Shown below is pseudo code representing the operation of the DSP software.

```
Initialize system
Wait till told to start
While not told to quit
{
    Gather audio data from codec
    Process audio data
    Report results
    Send audio data to TS-7200 over SPI
    Wait till reaction to reported results is complete
}
```

It is apparent from the pseudo code above that the DSP code is being commanded by another system. This represents the vehicle on which the system is installed. Once the system has been started and the audio data gathered and analyzed, the results of the analysis are reported to the vehicle, the data is logged on the TS-7200, and the system waits until the new heading is reached to sample the incoming audio data again. The communication to the vehicle is performed through a UART port. Since the 6416T has no native UART support, a software UART was implemented to interface with the vehicle. The software UART simply polls a digital I/O line watching for transitions at specified intervals that determine the baud rate of the serial line. It is currently set up for transmission at 115,200 baud, but any UART-supported speed is possible by changing the time interval.

The TS-7200 code operating on the Linux operating system is described by the pseudo code shown below.

```
Initialize system
While true
{
    Get Data packet
    Output amount of data received
    Store Data packet
}
```

This code implements the data logging for the system. The data logger software is designed to be turned on and run constantly until either the TS-7200 is shut down or until the user explicitly closes the software. The data is stored onto the compact flash card by this software for later retrieval.

5. SPI Transmission Protocol

SPI is a standard developed by Motorola for fast synchronous serial port communication. Systems using SPI transmissions are set up using a master-slave architecture. In the case of this project, the DSP is set up as the master and the TS-7200 is set up as the slave. This means that the DSP controls the speed of the transmission by sending its internal clock signal to the TS-7200. The transmission is accomplished using 5 wires: clock, frame, Master-In-Slave-Out (MISO), Master-Out-Slave-In (MOSI), and ground. One bit of data is transferred per clock cycle from the master to the slave and vice versa. Figure 3 shows a signal diagram of an SPI send.

The frame signal, labeled SFRMOUT/SFRMIN in Figure 3, remains high until a send is about to begin. Exactly one half-clock cycle after the frame signal goes low, the master puts the first bit of data onto the MOSI line that is labeled SSPTXD above. One half clock cycle after that, the clock signal is pushed high and the slave will extract the data from the MOSI line. Following the next half clock cycle, the slave will put its first bit of data, if it has any to send, onto the MISO line [5]. This continues until all bits in the current send are complete. Usually a single send will be 16 or 32-bits long depending on how the system is configured. Unlike other serial protocols, there is no added overhead to each send. Other protocols such as UART and TCP/IP add significant amounts of overhead to each byte sent, robbing data throughput capability. The SPI transmission protocol has no such drawback.

Not all SPI sends will look exactly like this. One of the advantages of the SPI standard is the ability to change the polarity of the signals, in effect making bit shifts happen at different points. It is also possible to add an additional bit delay to the start of each transmission from the point in time that the frame signal indicates that a send is about to take place. All this functionality makes SPI a versatile transmission protocol that can connect to many different sensors and systems.

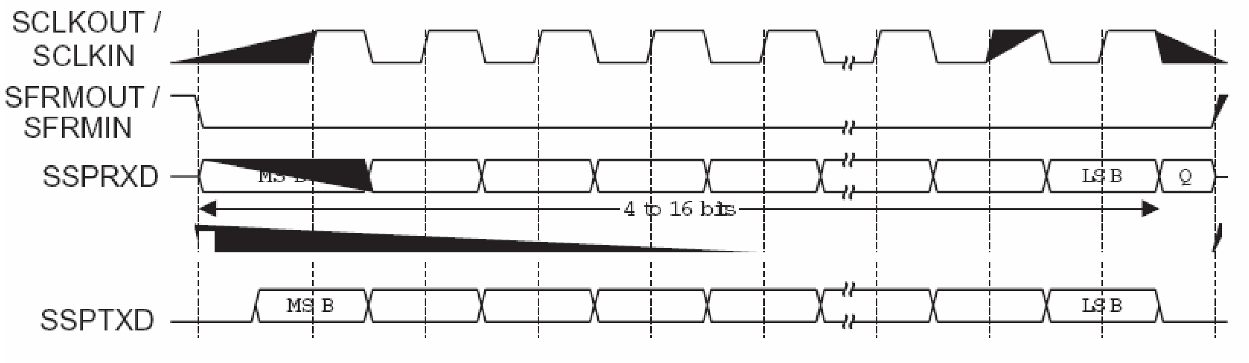


Fig. 3. Signal diagram of SPI send [5]

It is important to note that for the slave to send in a SPI transmission system, the master must be sending at the same time. The slave cannot simply decide it wants to send data but rather it must queue any data that needs to be sent to the master and when the master sends data the slave can simultaneously transfer the data that has been queued in its send buffer. Another important characteristic of SPI to note is that it provides no acknowledgment for the master or the slave to let the sender know that its data has been received. There is no guarantee of transmission quality.

6. SPI Transmission Limitations

The speed of an SPI transmission is dependent entirely upon the speed of the clock signal that the master uses. If the TS-7200 were to be used as the master, it could not sustain transmit rates of higher than 3.7 MHz or lower than 29 kHz because of the way the internal circuitry clocks the SPI system. The SPI port on the TS-7200 uses an internal signal of 7.4 MHz as its clock signal and uses an 8-bit divider to scale down that frequency for clock out. The 8-bit divider has a minimum value of 2 and maximum value of 256 yielding the frequency range of 29 kHz - 3.7 MHz [5]. The reason that the 8-bit divider has a minimum value of 2 is due to the nature of the SPI protocol. The SPI system must have knowledge of half bit times, and so the overall SPI system must have access to a clock that is at least two times faster than the clock speed of the transmission. This is accomplished by making the 8-bit divider equal to at least 2. The 6416T uses a 250 MHz signal as its base clock (System Clock / 4). Like the TS-7200, the 6416T also uses and 8 bit divider but it can range from 0 to 255 giving an effective range of 980.4 kHz to 250 MHz of available clock speeds [6].

The desired speed of the SPI system was determined to be 3 MHz giving a data transmission rate of 3 megabits per second. This would allow for 1 second of data recorded on the DSP to be sent to the TS-7200 to be stored on the compact flash card in 0.533 seconds.

The SPI transmission protocol was not designed for the application it is being used for in this project. It was designed primarily as a means of intraboard communication between components such as streaming sensors and microcontrollers. Attempting to use SPI to transmit data packets between two boards introduces problems. The electromagnetic interference that the signals will receive from other SPI signals may be great enough to disturb the transmission. This is minimized on a printed circuit board by placing the two components that are going to communicate using SPI very close to one another. This is not possible when communicating between two boards. Observing the SPI signals using a mixed signal scope, it was determined that when the clock signal and the data signal both went from a low state to a high state simultaneously, this would induce a voltage change on the frame line that would push it high enough momentarily to be registered as a logic high by the TS-7200. This essentially instructed the SPI system on the TS-7200 that the current frame being sent was complete, and so whatever data was not sent before the frame was mistakenly perceived to be toggled was lost.

Several measures were taken to reduce this affect. Wires connecting the two systems were made as short as possible, thereby reducing the effective surface area over which the signals could affect each other. A ribbon cable was used to make this connection for two other reasons that help to eliminate transmission problems. The SPI protocol is a clocked protocol and so any difference in the length of the wires used for transmitting could create propagation delays that may skew data by one or more bits. Using the ribbon cable helped to ensure this did not become a problem because the wires could all easily be cut to the same length. Also, ground lines were alternated between every signal line in the ribbon cable to help to minimize the interference each signal line could create on the other signal lines. The clock speed was also reduced to further reduce the amount of interference the between individual signals.

With these changes implemented, the system successfully communicates at speeds of up to 1.2 MHz, significantly less than the 3 MHz that was desired but still relatively fast. A speed of 1 MHz was eventually used for most system testing to ensure that electromagnetic interference was not a factor in system performance during the remainder of the project. This means that for every second of audio data that is recorded, another 1.6 seconds is needed to transmit that data onto the TS-7200.

7. TS-7200 SPI Limitations

The TS-7200 brings its own limitations to the transmission of data from the DSP. Whereas the DSP is setup to operate as a real-time system, meaning that when an event occurs that it needs to react to, it can immediately respond, the TS-7200 is not setup to perform this way. The Linux 2.4 kernel that was running on the TS-7200 is not a real time operating system nor is it even preemptible, meaning that even if an event is detected that needs to be addressed, the operating system cannot react to it until whatever process is currently occupying the processor releases it. The ARM microprocessor on the TS-7200 has an eight deep, sixteen bit FIFO (First in First Out) buffer for the SPI port that allows some room for storage of SPI data before data loss begins. However, this only allows for 128 microseconds of time for the processor to be interrupted away from the data logger program during a send before data is lost. Table 1 shows data gathered by sending 1 million 16-bit values from the DSP to the TS-7200.

Table 1. Data loss in transmission from DSP to TS-7200 with no operating system processes killed

Trial	Sent	Recvd	% Missed	Num ovflw	Avg miss per ovflw
1	100001	99994	0.0070%	1	7.0
2	100001	99991	0.0100%	2	5.0
3	100001	99995	0.0060%	1	6.0
4	100001	99974	0.0270%	4	6.8
5	100001	99977	0.0240%	2	12.0
6	100001	99995	0.0060%	1	6.0
7	100001	99989	0.0120%	2	6.0
8	100001	99991	0.0100%	3	3.3
9	100001	99995	0.0060%	1	6.0
10	100001	99989	0.0120%	2	6.0
Avg		99989	0.0120%	1.9	6.3

The collected data shows that over the course of a send of 100,000 16-bit values, only 0.012% of the data was missed during an average of only two overflows. This is not a significant amount of data to lose but is still not acceptable, primarily because the data is missed in consecutive samples. As discussed earlier, the operating system is running an FTP server as well as some other background services that are nonessential to the processor continuing to function. With those processes all stopped and removed from memory the performance of the SPI transmission of data from the DSP was significantly improved. Table 2 shows data gathered from sending 1 million 16-bit values from the DSP to the TS-7200 with all nonessential operating system processes stopped.

Table 2. Data loss in transmission from DSP to TS-7200 with all nonessential operating system processes killed

Trial	Sent	Recvd	% Missed	Num Ovflw	Avg Miss per Ovflw
1	100001	99985	0.0160%	2	8.0
2	100001	100001	0.0000%	0	0.0
3	100001	100001	0.0000%	0	0.0
4	100001	99984	0.0170%	1	17.0
5	100001	100001	0.0000%	0	0.0
6	100001	100001	0.0000%	0	0.0
7	100001	99974	0.0270%	4	6.8
8	100001	100001	0.0000%	0	0.0
9	100001	99984	0.0170%	1	17.0
10	100001	100001	0.0000%	0	0.0
Avg		99993.3	0.0077%	0.8	9.6

The system was able to transmit all 100,000 values with no loss on some occasions and the overall loss percentage has declined by thirty six percent. However, the average single overflow loss has increased in value making the localized problem of data loss even greater. It was determined that even this small amount of data loss was unacceptable. For this reason a transmission control protocol (TCP) was developed to provide a means of ensuring transmission success of all data sent from the DSP to the TS-7200.

8. TCP Details

The TCP designed to ensure data is successfully received by the TS-7200 is implemented differently on the slave and on the master side of the SPI port. On the master side, the DSP implementation of the TCP, data is sent in packets and then checked to make sure the entire packet was received. This is done by sending a special 16-bit value to instruct the TS-7200 to respond according to how many values it received since the completion of the previous packet. If the TS-7200 responds that it received all of the values that were sent in the packet, then the DSP continues to send the next packet. If the TS-7200 responds that it received either too many or too few values, the last packet is resent to ensure that the data is received correctly. After the last packet is sent and acknowledged the DSP will pause for long enough to ensure that the next send is received and then send a value instructing the TS-7200 that the current send is completed. For the slave implementation of the TCP, the TS-7200 simply counts how many values it receives as they stream in on the SPI port from the DSP. When the DSP sends the value that requests an acknowledgement of packet reception, the TS-7200 sends one value if all the values in the packet have been received and sends a different value if too many or too few values have been received. The TS-7200 then lines its data buffer up accordingly to receive what it expects will be the next packet of data sent. When the DSP sends the value that indicates that the current send is completed, the TS-7200 writes the data that has been received to the compact flash card and readies itself for the next block of data that will be sent.

The size of the packet that is used directly determines the amount of overhead that is incurred in the implementation of the TCP. In testing the TCP, the TS-7200 did not normally respond to the request for acknowledgement until the request had been sent approximately six times. This seems to indicate that the FIFO SPI buffer on the TS-7200 does remain somewhat full during sends. Using this value and the data collected earlier that indicated that approximately 1.9 overflows occur for every one hundred thousand 16 bit sends, the overhead can be statistically predicted for different packet sizes. Figure 4 shows the percentage of overhead plotted for packet sizes ranging from fifty to ten thousand.

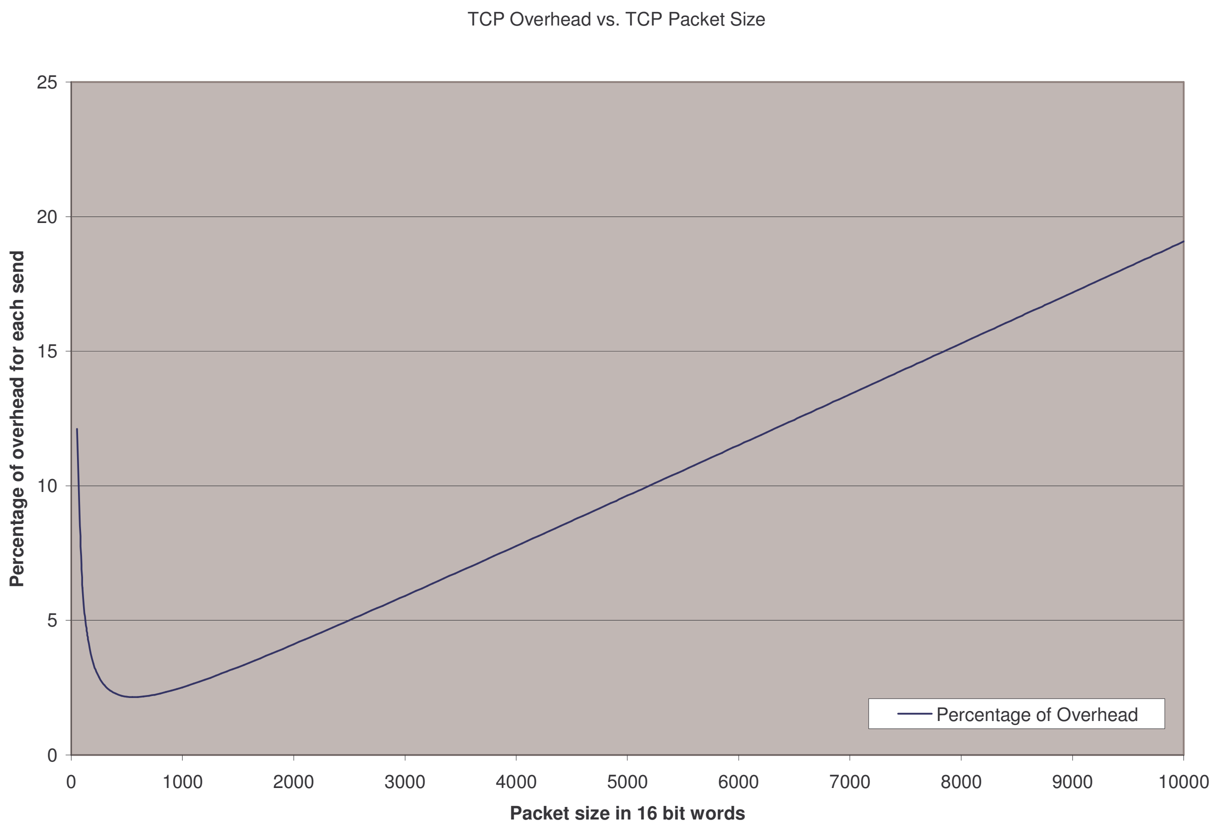


Fig. 4. Graph showing overhead incurred by different packet sizes

The optimal packet size was determined to be approximately five hundred and fifty 16-bit values that yielded an estimated overhead of 2.147%. The packet size was eventually set to one thousand though, which yielded only slightly worse performance at 2.5114% overhead. The higher number was chosen in the hopes that future upgrades to the system would reduce the number of overflows further which would make a larger packet size the better choice. If the average number of overflows per 100,000 16-bit values sent could be reduced to only one, then the choice of one thousand for a packet size is much closer to the optimal overhead solution than five hundred.

9. Overall System Performance

The TCP performs very well and after testing tens of millions of sends has yet to fail. This solution effectively overcomes all of the drawbacks of using the SPI transmission protocol. For every one second of audio data that is recorded, approximately two seconds of processing and logging time is needed. This means that every minute the system is operating twenty seconds of audio data is gathered, analyzed, and stored. System performance in terms of how much audio data can be used every minute could be increased significantly with an increase in the speed of the SPI transfer. The TS-7200 will eventually have to be mounted directly above the DSK6416T board for field testing inside of a five-inch watertight cylinder. This configuration will allow for pin-to-pin connectors to link the SPI ports on both systems, significantly minimizing the length of the connection, one of the factors limiting the speed of the transfer.

The system has not yet been deployed inside of an AUV for underwater testing but has performed well in laboratory conditions. Using a signal generator as its analog input, a significantly cleaner signal source than the hydrophone will eventually supply to the system, the system was able to record and analyze data exactly as designed. The algorithm run on the data was able to detect a difference from the carrier frequency with a resolution of one hundredth of one hertz. This is easily enough resolution to allow the system to determine the heading relative to the frequency source.

One of the other main requirements in the design of the system was to keep power consumption to a minimum. Using linear voltage regulators to power both the DSK6416T and the TS-7200, the overall power drawn to run the system is a little more than 7 Watts. However, a large part of that power is being dissipated as heat by the voltage regulators and is in effect wasted. Driving the hardware direct from a power supply takes only 4.5 Watts, indicating the voltage regulators are dissipating almost 3 Watts of energy as heat. The use of a more efficient switching power supply, as opposed to a linear voltage regulator, will take care of this problem.

10. Conclusions

DSP processors are very powerful and consume very little power, making them ideal for embedded applications. The massive amounts of data they are designed to handle and process however, is not matched by the data storage solutions that are available to them. Data logging for a DSP processor is possible to implement economically and efficiently. This paper details the design and implementation of an embedded DSP data logging system. The approach taken in designing this system is different than other DSP data logging solutions because a separate system is used to control and store the data rather than burdening the DSP with the details of implementing a file system and controlling a static storage medium. This frees the DSP to use most of its resources to process data that is what it is best at. The separate system is low power like the DSP and is able to transport the data off of the static storage medium over an Ethernet connection to a separate PC. An efficient and inexpensive ARM based single board computer running a Linux kernel was chosen because it met the requirements.

The system was able to meet the needs of the process and data intensive embedded DSP application detailed in this paper but has uses far beyond the original application it was designed for. Any DSP system that needs to log data in amounts that would be large or small could use this approach as an economical and versatile storage solution. This system could also be used as a stand-alone data logger. Many commercial data loggers exist that can perform the same functions that this design would be able to, but most require a full PC to run. This system could run using a fraction of the power and space of those more conventional data loggers. Other types of data loggers that are truly embedded do not have the storage space for audio data only sampling several thousand times before having to rewrite memory [7]. The fact that the DSP itself controls what data is logged without actually logging the data itself is also unique to this system.

If an analog to digital converter was used that was not geared towards the audio spectrum for data acquisition, this system could then be used to gather and store any type of analog data. This system would then stand up to several industrial applications by allowing it to communicate with large machines and monitor many processes simultaneously while storing them and sending that data to engineers who are attempting to optimize plant performance. Products already exist that fit this niche but could easily be made more cost effective with the use of this very inexpensive system [8].

11. References

[1] "Pentium III Processors: Overview", http://www.intel.com/design/intarch/pentiumiii/pentiumiii.htm, Dec. 2005.

[2] "TMS3206414T/15T/16T Power Consumption Summary", TI Online Application Notes, Dec. 9, 2005, http://focus.ti.com/docs/apps/catalog/resources/appnoteabstract.jhtml?abstractName=spraa45.

[3] "Applied Signal Processing: IDE Daughtercard", http://www.appliedsignalprocessing.com/idedaughter.htm, Dec. 9, 2005.

[4] "bf3Net TCP/IP Starter Kit", http://www.windmill-innovations.com/products/bf3Net/bf3Net_starterKit.htm, Dec. 2005.

[5] "Cirrus Logic: EP9301 Users Guide Release 1.00", February 2004, www.cirrus.com, Dec. 2005.

[6] "TMS320C6000DSP Multichannel Buffered Serial Port McBSP Reference Guide", September 2004, http://focus.ti.com/lit/ug/spru580d/spru580d.pdf, Dec 2005.

[7] C. Allan Morse, C. Richard Mummert, Rolando F. Martinez, Irving A. Gibbs, Edward C Prather, "New Multi-Processor Digital Excitation System", Power Engineering Society Summer Meeting, 2000, IEEE, July 16, 2000.

[8] Sakae Matsuzaki, Ariyoshi Tanzawa, Takashi Igarashi, Toyokazu Suzuki, Katsuhiko Tokushige, "Development of Data Logging System for Chemical Mechanical Polishing and Its Application for Process Control", IEEE Transactions on Semiconductor Manufacturing, Vol. 15, No. 4, November 2002.