Ultrasonic Imaging on the CML9S12DP256 Evaluation Board

Tyler Gilbert, c05tyler.gilbert@usafa.edu

EE 499, United States Air Force Academy

Abstract

The CML9S12DP256 evaluation board could possibly be used to implement an ultrasonic imaging interface. The DP256 does not have the processing power to do image processing, but the chip can be used to create pulses then sample the echo at 105KHz. The serial communications interface can relay the data back to a personal computer where the complex image processing can be done in MATLAB.

1. Introduction. The CML9S12DP256 microcontroller board was used along with MATLAB to implement a phased array ultra-sonar, imaging system. Four elements were used for transmitting and four for receiving. The transducers used are JAMECO part #136653. A phased array sonar device operating at ultrasonic frequencies provides much better resolution than sonic frequencies. However at higher frequencies, the higher resolution is traded for a decrease in range. Higher frequencies also require a system with more capability. The DP256 had several shortcomings in implementing the ultra-sonar system. The sampling rate was limited to 105KHz by the clock speed. The serial communications interface was also limited due to the clock. The figure below shows a picture of the prototype.

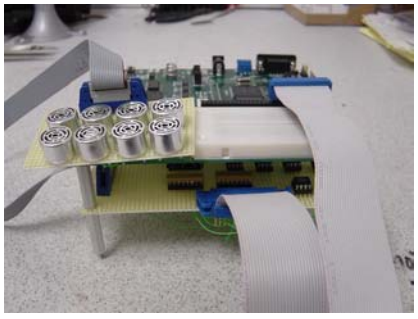

Figure 1. Picture of Prototype

2. Theory. Phased array sonar has the ability to detect range and azimuth. This information can then be fused to create a two-dimensional image. To detect the range is very simple: measure the amount of time it took the sound to travel to and from the destination and multiply by half the speed of sound.

$$d = a\frac{t}{2}$$

Equation 1. Ranging Equation

The azimuth is determined by measuring the phase difference between the receivers and can be complimented by steering the transmitted beam in the direction of interest.

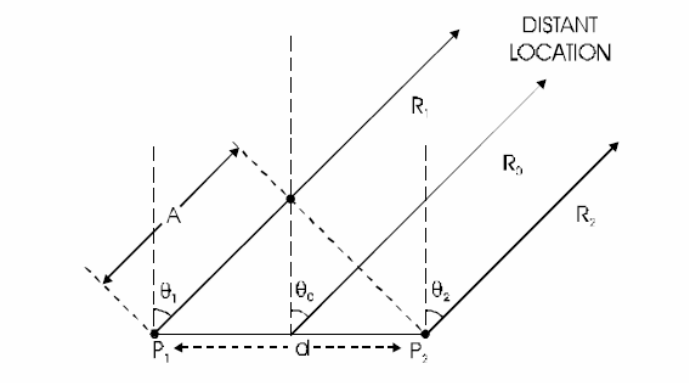

Figure 2. Detecting Azimuth [1]

In the figure P_1 and P_2 are the receivers. The far-field assumption allows θ_1 and θ_2 to be equal but requires that R_1 and R_2 are much greater than d, the spacing between the receivers. The incoming angle is determined by A which is the difference in distance between R_1 and R_2 found by taking the time delay multiplied by the speed of sound. Using trigonometry θ_1 can be found using the following equation.

$$90 - \theta_1 = \cos^{-1}\left(\frac{A}{d}\right)$$

Equation 2. Calculating Azimuth [1]

This method of azimuth calculation is coupled with beam steering. Using multiple transmitters will cause specific construction patterns depending on the spacing between the transmitters and the time delay of the output waveform. The ideal spacing between the transmitters is half-wavelength. The beam pattern of half-wavelength, spaced omni-directional transmitters is shown below.

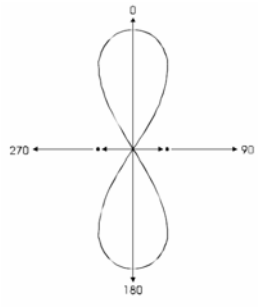

Figure 3. Half-wavelength Interference Pattern [1]

The following equation shows how to calculate the angle of the main construction lobe.

$$\frac{d}{\lambda}\sin\theta = 0,1,2,3...$$

Equation 3. Interference Angles [1]

The destruction occurs where the equation is equal to 0.5, 1.5, 2.5, etc. If the spacing is greater than half-wavelength, the construction pattern results in large side lobes next to the main lobe. The figure below illustrates the 1.5 wavelength spacing contrasted to half-wavelength spacing.

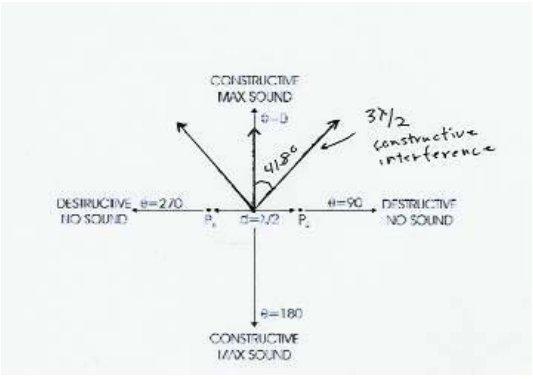

Figure 4. Non-standard Spacing Interference [1]

The original figure shows the constructive and destructive patterns of half-wavelength spacing while the hand-written overlay shows several of the 1.5 wavelength constructive pattern. If the spacing is more than a half-wavelength, beam steering is quite challenging. Instead of steering a nice single beam, the system has to deal with three beams. However, the receivers should still be able to filter out the results along the correct beam as long as there is a beam in the direction of interest. The main beam is steered by varying the time delay to each of the transmitters.

By delaying the pulses to each transmitter, the constructive interference pattern steers to the left or the right. Consider three transmitters distance, d, apart.

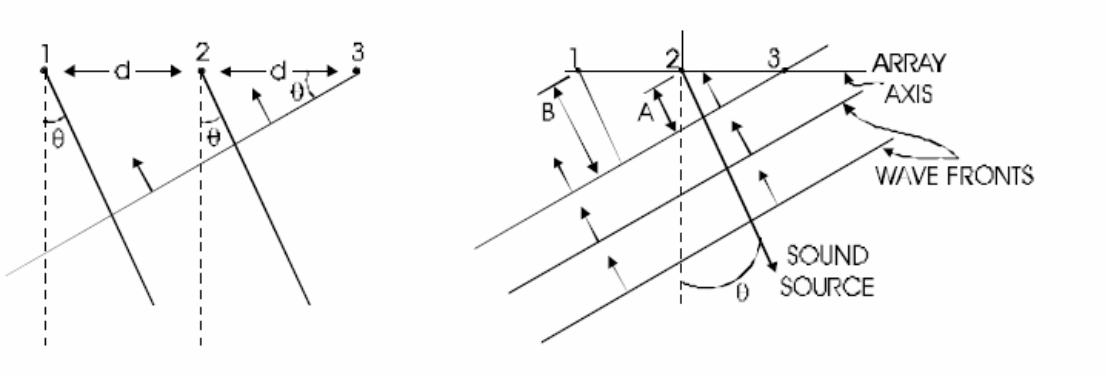

Figure 5. Beam Steering Angle [1]

The wave fronts should all be aligned in the desired steering direction. Again using trigonometry:

$$A = d \sin\theta$$
$$B = 2d \sin\theta$$

Equation 4. Change in Distance [1]

To get the respective time delays for each transmitter simply divide by the speed of sound, a.

$$t_1 = \frac{A}{a}$$
$$t_2 = 2t_1$$

Equation 5. Time Delays

For a four element phased array, the transmission delays would be 0, t_1, $2t_1$, and $3t_1$. Consequently, for a given time delay of t1, the corresponding beam steering angle would be:

$$\theta = \sin^{-1}\left(\frac{t_1 a}{d}\right)$$

Equation 6. Beam Angle

The is a summary of the basic function of a phased array sonar implementing beam steering to increase the sensitivity of the system along the desired beam angle.

3. Design. The main platform for handling beam steering and sampling is the CML9S12DP256 microcontroller board or DP256. The outgoing pulses are connected to an output port and amplified using the 754410 amplifier integrated circuit (IC). The incoming data is amplified using a Burr-Brown rail to rail operational amplifier, OPA2340. The amplified signal is then sampled using Texas Instruments' ADS7816 serial analog to digital converters (ADCs). The ADCs can operate as high as 200KHz, but the microcontroller limits the sampling rate to 105KHz. The microcontroller passes the data back to MATLAB via the serial communications interface.

The DP256 is interfaced with MATLAB using the serial communications interface (SCI). The SCI is not fast enough to support a real-time system. The theoretical maximum for the SCI port is 500,000 baud. However, no computer supports this speed. The fastest practical baud rate is 57600 baud, which is achieved by setting the SCIBD register to 0x09. More detail on implementation are discussed later.

The DP256 is interfaced to the sampling board using a 26-pin ribbon connector. The following figure shows the pin descriptions of how the board is interfaced to the sampling circuitry. It also shows the pin description of the sampling circuitry and the ultrasonic

transducers. The ultrasonic transducer board could be replaced with transducers that operate at lower frequencies.

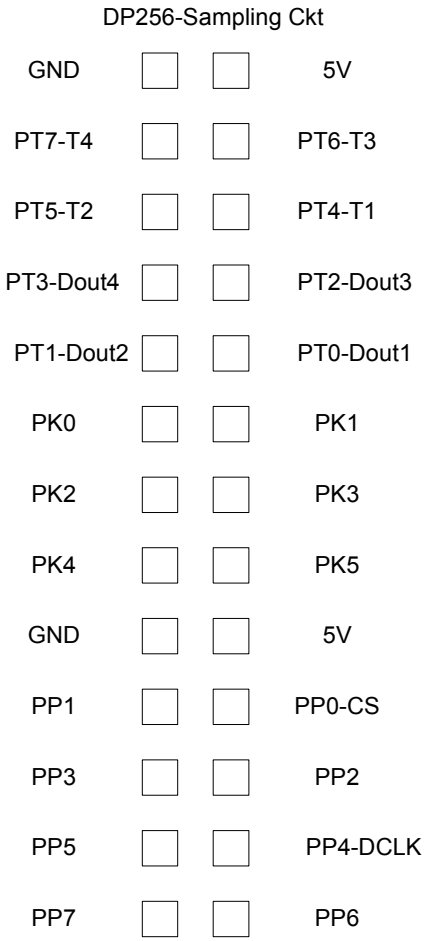

Figure 6. DP256 to ADC Board Interface

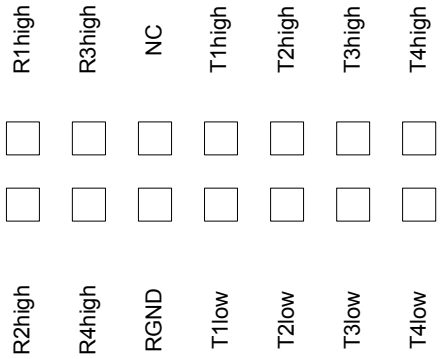

Figure 7. ADC Board to Transducer Interface

The chip select and data clock are the same signals for all four of the ADCs. The receivers have a common ground which should be connect to 2.5V. The transmitters do not have a common ground but both inputs are connected to the output of the 754410 IC.

The operational amplifiers and analog to digital converters are wired as shown in the figure below. The OPA2340 is a dual rail to rail operational amplifier (OPAMP). It comes within about 200mV of each input. Regular 741 OPAMPs will only come within about 1V or so of the inputs. The circuit is set up so that the transducer receivers are connected to a virtual ground at 5V. The virtual ground is created using a simple voltage divider.

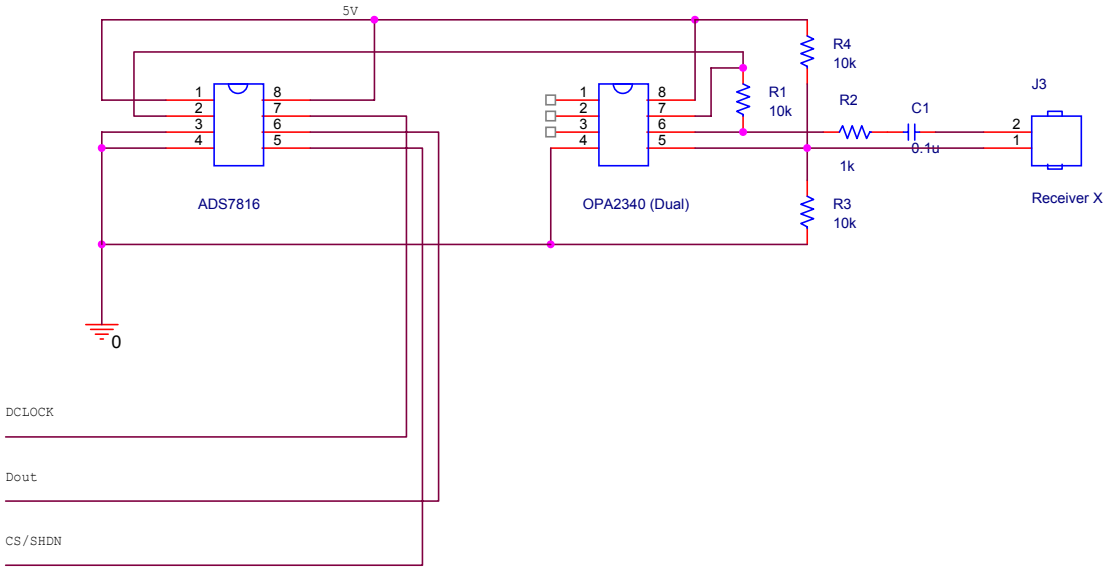

Figure 8. ADC Circuit

The waveform is amplified to oscillate about 2.5V and then fed into the ADCs. One big advantage of using these external analog to digital converters is that the microcontroller can sample up to eight channels at the same time. In this case, all four conversions are made at the same time. It does however take extra processing power to convert the data to a usable form.

The transmitters are connected to the 754410 which is a very simple way to amplify the output. By using a not gate in conjunction with the 754410, the input power turns into the peak measurement rather than the peak-to-peak value ergo doubling the power output.

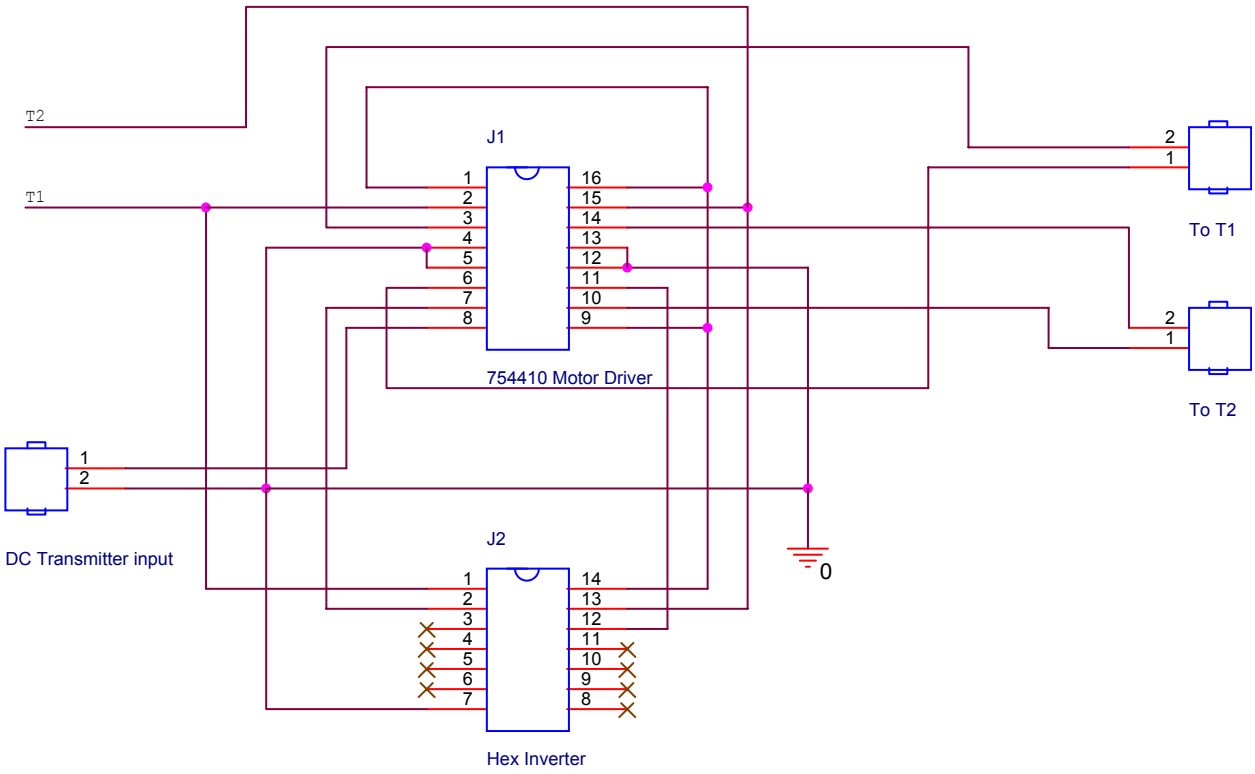

Figure 9. Transmitter Circuit

Finally, the transducers are spaced 2.1-wavelengths apart. Due to the physical size of the transducers, this is the closest they could possible be placed. T1 and R1 are on the right in the figure below so that from behind the system T1 and R1 are on the left.

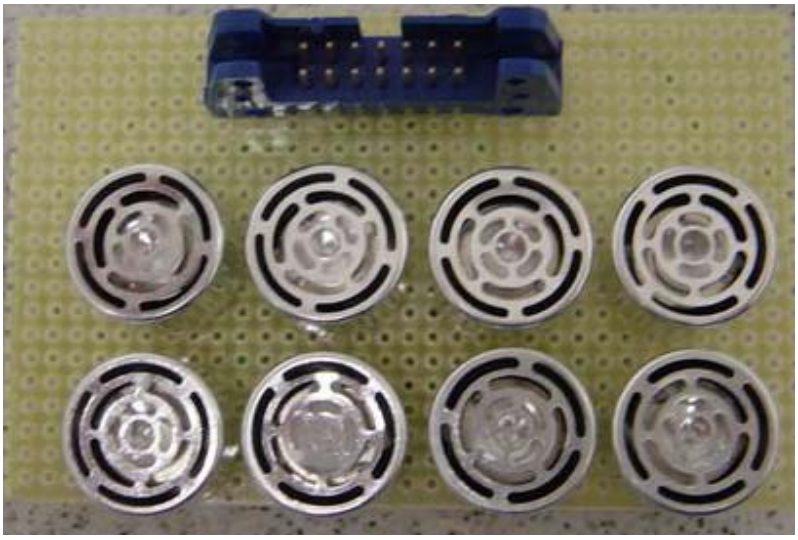

Figure 10. Transducer Setup

This spacing constraint has implications on the operation of the system. The beam cannot be steered very precisely due to the multiple lobes. The implications are similar to figure X, which shows 1.5-wavelength spacing. The following figure simulates the wave fronts of the 2.1-wavelength spacing next to half-wavelength spacing.

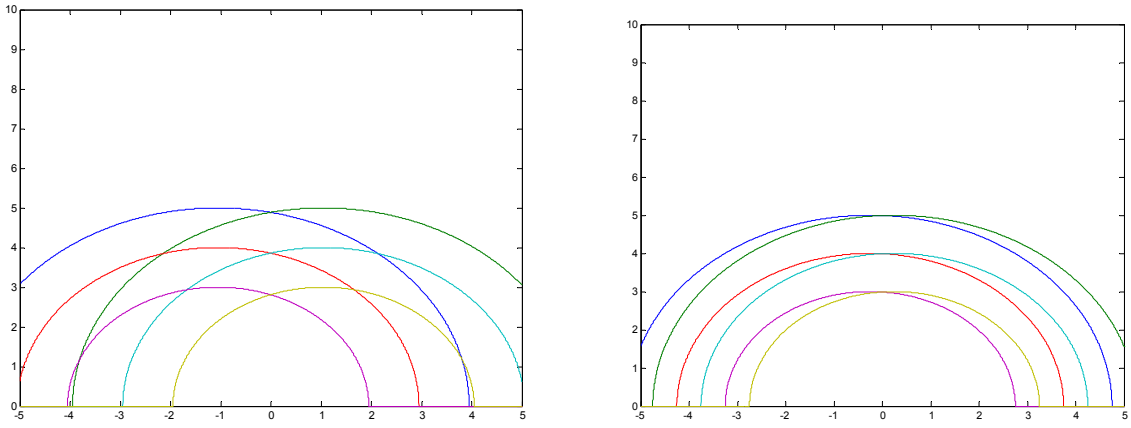

Figure 11. 2.1 Wavelength Spacing (left) and Half-wavelength Spacing (right)

The half-wavelength spacing has construction at 0 degrees and total deconstruction at 90 degrees. However, the 2.1-wavelength spacing is not as clean. The main constructive

interference occurs at 0, and plus or minus 28 degrees. When beam steering, the same pattern is steered decreasing the relative sensitivity in the desired direction.

4. Implementation. The implementation of this design consisted of triggering the DP256 to send out transmission pulses, sample the echo and send the data back through the serial port back to MATLAB. MATLAB could then use the data to create an image.

The DP256 runs in an infinite loop where it is constantly responding to requests from MATLAB. MATLAB sends the desired beam steered angle to the DP256. The DP256 then transmits in the indicated direction and sends the echo back to MATLAB. The following figure shows the software flow diagram of the DP256 program.

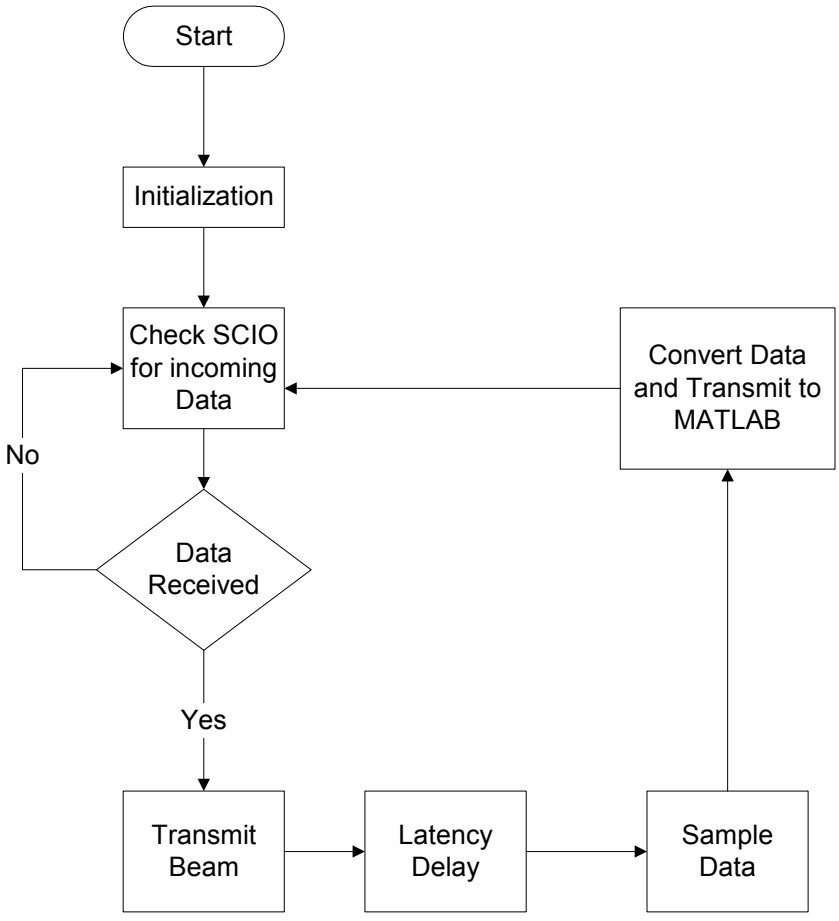

Figure 12. DP256 Flow Diagram

The incoming data determines where the beam should be steered. The delay accounts for the propagation delays in the ADCs and OPAMPs and should account for the length of the transmission. The delay can also be lengthened to listen to objects that are farther away. By taking 256 samples at 105KHz, the system has a window of 14.63 inches. This number is found by first determining that sound travels 114mils between samples. After 256 samples, the sound has traveled 29.26 inches. Divide 29.26 inches by two because the sound has to travel the distance twice—to the object and back.

The data is sampled at a frequency of 105KHz. This number is limited by the speed of the processor. The code is written in assembly to absolutely optimize the sampling rate. The ADS7816 is a 12-bit A/D converter, but to get over the Nyquist rate the DP256 only grabs the 8 most significant bits. The figure below shows the timing scheme for the getting data from the ADS7816.

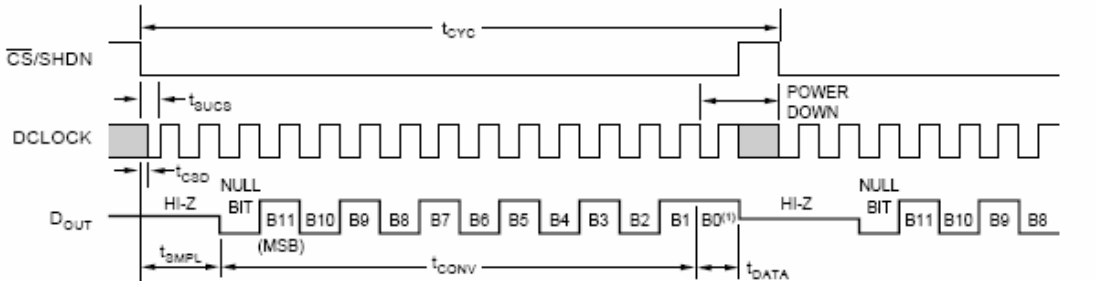

Figure 13. ADC Timing Diagram [2]

The DCLK is connected to the pulse width modulator (PWM) on the DP256. By using the PWM unit, the DP256 does not have to use processing power to oscillate the clock. The function to get samples first turns on the PWM to oscillate ever six clock cycles. There are some no ops inserted in order to account for t_{SMPL}. By using the "movb" instruction in indexed addressing mode, the DP256 reads a byte from port T and stores the value in an array holding raw data. The

sampling frequency is determined by the time between the falling edges of the chip select line which is about 9.52μs. The condensed code is shown below which compiles using ICC12.

```
        input = &raw_input[0][0];  //points input to the raw_input array
        asm("ldy _input");      //points y to the raw input array (256 bytes)
        asm("ldaa #255");       //gets register a ready to take 256 samples
asm("Loop256:");
        PORTP = CS_LOW;         //clears Port P bringing the chip select line low
        asm("ldab #3");         //sets the duty cycle of PWM4 to 50%
        asm("stab 0xC0");        //0xC0 is PWMDTY4
        NOP();
        …                       //needs 18 NOPs (two shown)
        NOP();
        asm("ldab #5");         //This gets register b ready set the PWM (DCLK) to 100%
       asm("movb 0x240,1,y+");  //Execute this instruction 8 times total (0x240=PORTT)
       …                        //needs 7 movb's here
       asm("movb 0x240,1,y+");
       asm("stab 0xc0");        //This  sets the duty cycle to 100% (always high)
       asm("movb 0x240,1,y+");  //The last movb occurs after the previous instruction
       PORTP = CS_HIGH;         //Sets the chip select high
       asm("deca");             //loops until the routine grabs 256 samples
       asm("bne Loop256");
       asm("sty _input");       //points input to the end of raw_input
```

Once the data is sampled it needs to be converted to bytes and sent to MATLAB. As the data stands the 8-bits for each receiver are spread across the four least significant bits of 8-bytes.

The converting time is small compared to the time needed to transmit the data through the serial port. In order to maximize efficiency, the DP256 can convert the data while the serial port is sending data to MATLAB.

Once the serial port has been initialized, writing data to MATLAB is quite simple. If the DP256 board is running under MON12, the MON12 debugging program will initialize the hardware timers on the chip automatically, including setting the SCI0 baud rate to 9600. The manufacturer provides the details for running the chip in stand-alone mode. The baud rate depends on the SCIBD register. By writing 0x09 to SCIBD, the baud rate will be 57600. This is the fastest baud rate that will work with the PC's standard baud rates. Once the port is

initialized, a write to the SCIDRL register while the SCI is idle will initiate a transfer. Programming the serial port with MATLAB is also relatively simple.

MATLAB uses the "serial" function to set up the serial communications port. The "fopen" function opens the port while "fclose" closes the port. However, "fclose" will not free the serial port so that it can be used be other programs. The "freeserial" function must be used to allow other programs access to the serial port after MATLAB is done reading and writing to it.

```
N = 255;
s = serial('COM1','BaudRate',57600,'Parity','none','InputBufferSize',N*4);
fopen(s);
fprint(s,value);        %the "value" written determines the
new_data = fread(s);  %reads 255*4 values from the serial port
…                       %plot the data, apply DSP techniques
fclose(s);
freeserial;
```

The data that is returned to MATLAB are four waveforms received from each transducer. Each waveform is 256 bytes long. At 57600, it takes 178ms to transfer all 1024 bytes. In addition it takes another 10-15ms for the DP256 to send out a steered beam and receive the data through the transducers. Ergo the best refresh rate for the image processing will be about 5Hz. Because of the mechanical properties of the transducers, the incoming data experiences a lot of ringing.

When the transducers are hit with a high voltage, 25Vp, the transmitters take some time to dampen out. Because of the attenuation the receivers only produce a small voltage so the dampening effects are less on the receiving end. The OPAMPs amplify the signal 10 times. So the largest voltage the receivers are designed to produce is about plus and minus 250mV to accommodate for the full 5V range of the rail to rail OPAMPs. The following figure shows a reflected wave off of a metal object about 3, 6, and 9 inches away from the transmitter. The transmitter was hit with a 8 pulse wave at 25Vp.

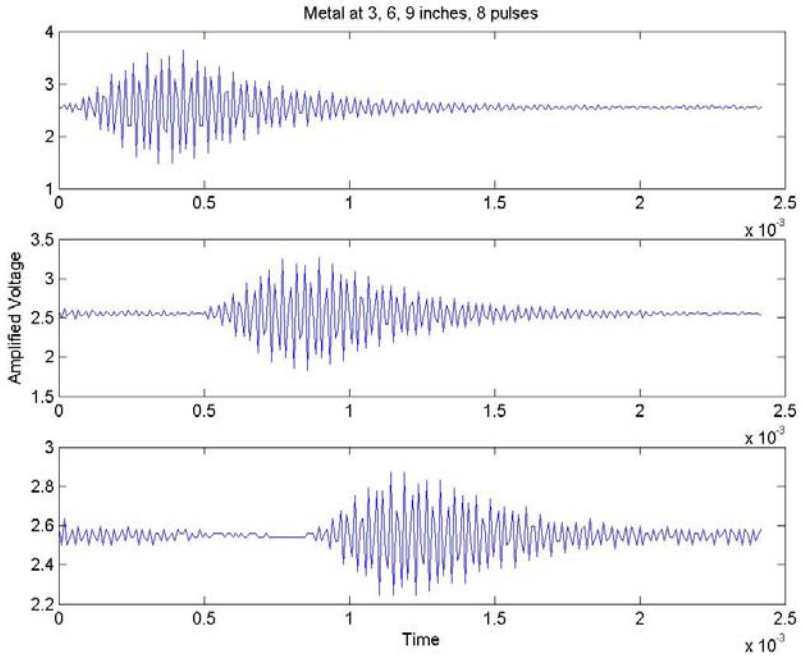

Figure 14. Ranging Data

The initial noise seen on the third graph was due to a serial port reading problem. The data from the last samples was overlapping to the first samples. This problem was later fixed by adjusting the input buffer size. The majority of the incoming data is very near to 40KHz, which is what the transducers are designed for.

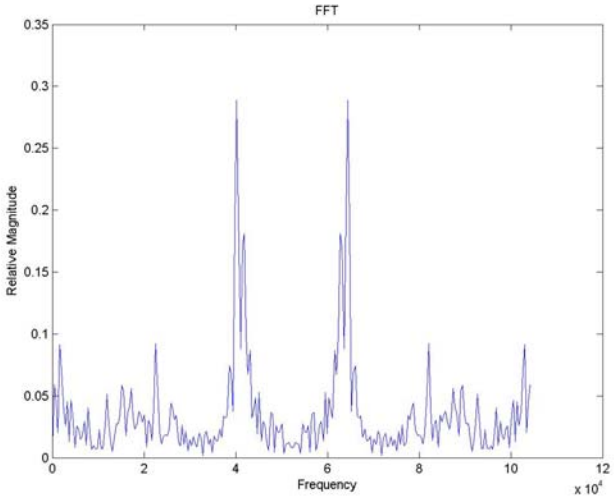

Figure 15. Frequency Spectrum

The sampling frequency is 105KHz. The main peak is at 40KHz, and the mirror peak is over the $\frac{F_s}{2}$ point so that it mirrors back onto 40KHz. The figure below shows four sampled waveforms overlaid on the same plot. The figure was generated using the files XXX.c and Incoming.m.

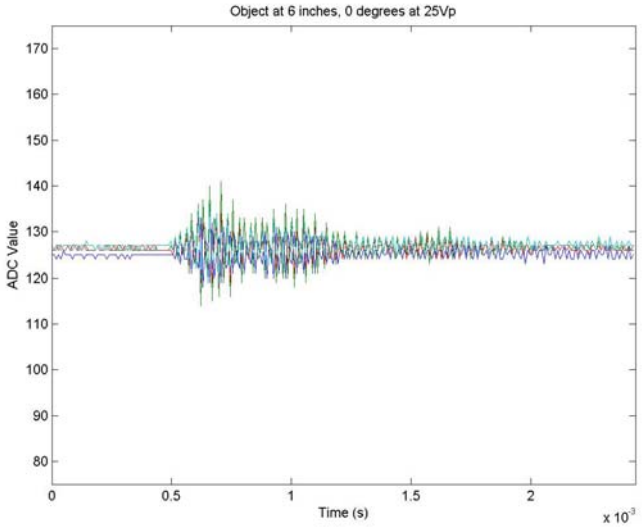

Figure 16. Four Data Streams, Object at 6 inches

The object was six inches away and started returning a signal after 500ms of collecting data.

One more plot helps to calibrate the range finding ability of the system.

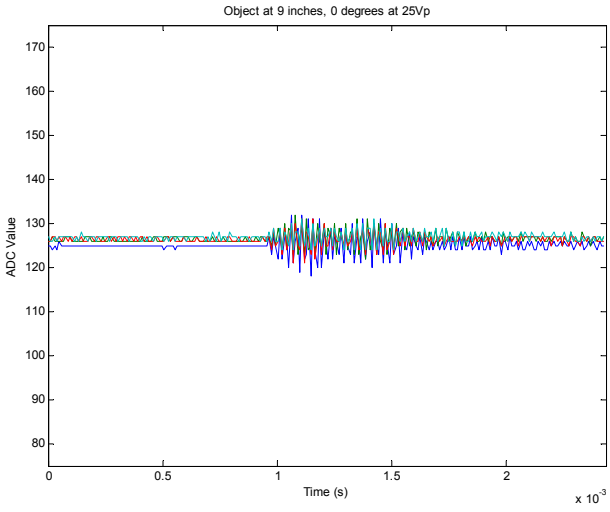

Figure 17. Four Data Streams, Object at 9 inches

At nine inches it takes the signal 1ms to return a signal. This means the sound traveled 3 inches * 2 in 500ms. The preceding numbers yield a local speed of sound of about 1000ft/s which is about right.

The data sampling meets the Nyquist rate. The four beams do not have any delays implemented to the constructive interference pattern follows figure X above. The object was placed offset 30° to the right. From the waveforms, the angle could be calculated by measuring the phase shift between the waveforms. All things considered the receiver circuit works relatively well, the beam steering mechanism is however lacking in several ways.

Two pieces of hardware make it difficult to effectively steer the outgoing beam with this system at this frequency. The physical spacing of the transducers coupled with the transmitting width makes beam steering shaky. The transducer transmits ultrasonic energy at about plus and minus 30 degrees. Considering the constructive interference patterns inherent to the physical spacing of these transducers produces three main beams (see figure X) in this range, beam steering is not practical with the system.

Nonetheless, the DP256 has two potential solutions for producing the desired waveforms in order to steer the beam either using the output compare module or the general I/O ports. To beam steer each transducer should start delayed from left to right or right to left depending on the desired angle. The DP256 produces square waves which are mechanically filtered by the transducers. The delays are dictated by equation X above. After the delay, the DP256 should output a 40KHz square wave. This turns into toggling the line every 100 clock cycles. The resolution of each beam angle depends on how small the DP256 can make the delays. Using the output compare module produces erratic results when the DP256 tries to make the delays very small. By pre-calculating the delays, the DP256 could theoretically use interrupts to toggle the

output which would allow the system to change the output values every clock cycle. However, usually one of the four output compare events will inevitably happen when another event is being serviced by the interrupt. This will cause the other event not to be serviced. Hence, it would be better to implement the general I/O ports rather than the output compare.

The delay resolution using general I/O would be determined by how fast the DP256 can move bytes from pre-calculated waveforms in memory to the register associated with the desired port. From the get samples function, which uses a similar approach, the fastest the DP256 could output a new value is about every 6 clock cycles. This would translate into a transmitting with a time delay of 750ns. Using equation X above, the beam angle resolution is 0.8°. The drawback is that the pre-calculated waveforms would take a lot of memory.

The waveform requires a byte every 6 clock cycles. To oscillate at 40Khz, a pulse takes 200 clock cycles every period. With no delays, it would take 1066 bytes for one beam angle profile. Steering at plus and minus 30° with 0.8° resolution would take 75 beams. That equates to over 75KB of memory. The DP256 does not have 75KB of RAM. So the profiles would have to be programmed into the 256KB of FLASH memory. Because the transducers do not physically support beam steering, the idea is moot. However, the same principles could be applied using different transducers that would meet the spacing requirements. The system connects with the transducers using a 14 pin port illustrated in figure X above which could easily be switched with a different transducer board.

5. Conclusion. The DP256 is not the best platform for a high performance digital signal processor. The DP256 would do much better implementing a sonar system at lower frequencies. The transducers had a fatal pitfall due to physical size. Half-wavelength spacing helps to simplify a very complex problem. Because of the constructive interference pattern caused by the

spacing, the transducers do not practically support beam steering. Nonetheless, beam steering is not a necessary function of the system. The system could still produce an image by processing the data received by listening to a pulse train transmitted in all directions.

The rail to rail OPAMPs are a very good component for systems that are limited to a single 5V source. It is very easy to create a virtual ground at 2.5V around which the incoming data can oscillate. They function well with the analog to digital converters which also operate with a 5V source.

External serial ADCs are a good way to make a major improvement on the on board DP256 ADCs. The DP256 has two ADCs which operate as fast as 142KHz. However, there are only two ADCs which means that the sampling frequency for four channels is only 71KHz which is below the Nyquist rate for 40KHz ultrasonic waves. The serial ADCs were simple to implement and increased the total sampling rate to 105KHz for up to 8 channels.

The best implementation of this system would be to attach transducers that could use half-wavelength spacing on both transmit and receive. Another shortfall of the system is that the serial communications port does not practically support the sampling frequency of which the DP256 is capable. Any similar system would benefit by having a higher speed data transfer implementation such as USB or parallel port.

The current system lacks the capability to steer the beam. Nonetheless, it meets the Nyquist sampling rate and is able to report 1KB back to MATLAB in approximately 200ms.

Works Consulted

[1] "Multibeam Sonar Theory of Operation." Communications SeaBeam Instruments, 2000.
[2] Texas Instruments ADS7816 datasheet.
[3] Burr-Brown OPA2340 datasheet.
[4] 754410 Motor Driver Circuit Datasheet.