

Jen Grier  
DST: Final Project  
Chad Wagner  
December 12<sup>th</sup>, 2007

### **Convolution on the Atmega168 Chip using Arduino**

This project enables an Atmega168 chip to convolve either one or two analog inputs to either a printed value or a control for a motor, LED, or other electronically-controller devices using the Arduino language. Arduino is an open-source language roughly based on the C library. The attached CD-ROM includes the Arduino programming environment along with the two .pde files that contain the code. I engaged this project because of it's implications in digital signal theory as a fundamental and incredibly useful manipulation of audio data that could have unique implications in other digital applications.

The similar components of both version of my program are as follows: Firstly, the variables are set beforehand. Arrays are set to hold a specific amount of values for the initial convolution, the impulse, and the final resulting array. Data inputs are set to the correct ports (in the case of the Atmega168, these are called pins). Also, count and storage position values were set to count, count2, and storagePos to

*E85.2599-Fall-2007-jlg466*

assist the for loops in creating an increasing timer to increase the array values appropriately (not unlike the for k:10 value often seen in our Matlab assignments).

This is followed by the first loop that the begins the run environment on the chip, known as the setup loop. During this, the rate of communication needs to be set (I set it to 9600 baud, which is easy for serial communication between a computer and this chip). Also, the declared pins for collecting values needed to be set to understand the incoming data. I used analog inputs and outputs because the range of numbers would be flexible towards use with all kinds of physical sensors or audio devices.

The processing of data occurs in the void loop. In the first version, the for loop collects values from the pin assigned to convolution input, and is set to do so ten times. In the second version, there are two similar loops to collect values both for the impulse and the initial convolution input.

The second loop uses two variables to count the amount of times the multiplication and addition processes, respectively, occur. By using these two variables, the addition process can move to the appropriate place for each pairing of convolution and impulse values, with an outer storagePos variable that

*E85.2599-Fall-2007-jlg466*

increments until 19, which is the limit of the 10 x 10 convolution this program is set to do.

Finally, I have the resultArray set to print back to the Arduino environment via serial connection, but this feature does not work without the Atmega hardware to commune with. Depending on the application, these values could easily be set to send to an LED array, motor controls, wireless communication, or a variety of other physically interactive components with ease.

A hard copy of the code follows for both iterations. Feel free to redistribute as you like.

References:

<http://Arduino.cc>, the main online reference for Arduino code and tutorials

E85.2599-Fall-2007-jlg466

```
/* Jen Grier (jlg466@nyu.edu)
 * N14007763
 * December 13th, 2007
 *
 * DST Final: Applying Signal Theory to Arduino: Convolution
 * Version I: Preset Impulse Convolvering an Analog Input
 *
 * Thanks to David Cuartielles for his Knight Rider tutorial on
Arduino.cc!
 * Thanks to Tyler Worman for checking my for-loops!
 */

int myImpulse[] = {1, 0, 1, 6, 7, 5, 8, 9, 0, 4}; //Demo impulse
for this program
int convolvePin = 1; //Where the convolution values come from
int answerPin = 2; //Output of convolution result
int conVar; //Where the initial convolution values are read
int conArray[10]; //Sets the conVar as an array to be filled
with 10 values
int resultArray[19]; //Sets the resultArray as an array to be
filled with 19 values
int count = 0; //For moving around the for-loop with the conv
array
int count2 = 0; //For multiplying with the
int storagePos = 0; //This gives me a position to move around
for addressing the array, value by value

void setup() {
  Serial.begin(9600); //Sets speed of conversation
  pinMode(convolvePin, INPUT); //Sets the input for the
convolution values
  pinMode(answerPin, OUTPUT); //Sets the output for the result
values
}

void loop() {
  for (count=0; count<=9; count++) { //For each step in the
counter up to 10, grab a value for conArray
    conVar = analogRead(convolvePin); //Reads the convolution
values
    analogWrite(conVar, conArray[count]); //Writes value to
proper place in the array
  }
  for (count=0; count<=9; count++) { //The convolution loop
```

E85.2599-Fall-2007-jlg466

```
    for (count2=0; count2<=9; count2++) { //A subloop to do the
multiplication per each value of myImpulse
        resultArray[storagePos + count2] += (conArray[count] *
myImpulse[count]);
    }
    storagePos++; //This moves the process over per each value
in the result array
    if (storagePos = 19) { //Stops the convolution loop when the
storage position moves to 19 (nothing more to convolve!)
        break;
    }
}
// analogWrite(answerPin, resultArray); //Sends result as
analog values
Serial.print(resultArray[1]); //Prints answer as ASCII
Serial.print(resultArray[2]); //Prints answer as ASCII
Serial.print(resultArray[3]); //Prints answer as ASCII
Serial.print(resultArray[4]); //Prints answer as ASCII
Serial.print(resultArray[5]); //Prints answer as ASCII
Serial.print(resultArray[6]); //Prints answer as ASCII
Serial.print(resultArray[7]); //Prints answer as ASCII
Serial.print(resultArray[8]); //Prints answer as ASCII
Serial.print(resultArray[9]); //Prints answer as ASCII
Serial.print(resultArray[10]); //Prints answer as ASCII
Serial.print(resultArray[11]); //Prints answer as ASCII
Serial.print(resultArray[12]); //Prints answer as ASCII
Serial.print(resultArray[13]); //Prints answer as ASCII
Serial.print(resultArray[14]); //Prints answer as ASCII
Serial.print(resultArray[15]); //Prints answer as ASCII
Serial.print(resultArray[16]); //Prints answer as ASCII
Serial.print(resultArray[17]); //Prints answer as ASCII
Serial.print(resultArray[18]); //Prints answer as ASCII
Serial.print(resultArray[19]); //Prints answer as ASCII
delay(500); //Tells the chip to wait for a half-second
}
```

E85.2599-Fall-2007-jlg466

```
/* Jen Grier (jlg466@nyu.edu)
 * N14007763
 * December 13th, 2007
 *
 * DST Final: Applying Signal Theory to Arduino: Convolution
 * Version II: Analog Impulse Convolving an Analog Input
 *
 * Thanks to David Cuartielles for his Knight Rider tutorial on
Arduino.cc!
 * Thanks to Tyler Worman for checking my for-loops!
 */
```

```
int impVar; //Impulse variable for reading pin
int impulsePin = 2; //Sets pin for reading impulse values
int myImpulse[10]; = //Storage for impulse array values
int convolvePin = 1; //Where the convolution values come from
int answerPin = 3; //Output of convolution result
int conVar; //Where the initial convolution values are read
int conArray[10]; //An array to be filled with 10 conVar values
int resultArray[19]; //Sets the resultArray as an array to be
filled with 19 values
int count = 0; //For moving around the for-loop with the conv
array
int count2 = 0; //For multiplying with the
int storagePos = 0; //This gives me a position to move around
for addressing the array, value by value
```

```
void setup() {
  Serial.begin(9600); //Sets speed of conversation
  pinMode(convolvePin, INPUT); //Sets the input for the
convolution values
  pinMode(impulsePin, INPUT); //Sets the input for the impulse
values
  pinMode(answerPin, OUTPUT); //Sets the output for the result
values
}
```

```
void loop() {
  for (count=0; count<=9; count++) { //For each step in the
counter up to 10, grab a value for conArray
    conVar = analogRead(convolvePin); //Reads the convolution
values
    analogWrite(conVar, conArray[count]); //Writes value to
proper place in the array
```

E85.2599-Fall-2007-jlg466

```
    }
    for (count=0; count<=9; count++) { //For each step in the
counter up to 10, grab a value for conArray
        impVar = analogRead(impulsePin); //Reads the convolution
values
        analogWrite(conVar, myImpulse[count]); //Writes value to
proper place in the array
    }
    for (count=0; count<=9; count++) { //The convolution loop
        for (count2=0; count2<=9; count2++) { //A subloop to do the
multiplication per each value of myImpulse
            resultArray[storagePos + count2] += (conArray[count] *
myImpulse[count]);
        }
        storagePos++; //This moves the process over per each value
in the result array
        if (storagePos = 19) { //Stops the convolution loop when the
storage position moves to 19 (nothing more to convolve!)
            break;
        }
    }
}
// analogWrite(answerPin, resultArray); //Sends result as
analog values
Serial.print(resultArray[1]); //Prints answer as ASCII
Serial.print(resultArray[2]); //Prints answer as ASCII
Serial.print(resultArray[3]); //Prints answer as ASCII
Serial.print(resultArray[4]); //Prints answer as ASCII
Serial.print(resultArray[5]); //Prints answer as ASCII
Serial.print(resultArray[6]); //Prints answer as ASCII
Serial.print(resultArray[7]); //Prints answer as ASCII
Serial.print(resultArray[8]); //Prints answer as ASCII
Serial.print(resultArray[9]); //Prints answer as ASCII
Serial.print(resultArray[10]); //Prints answer as ASCII
Serial.print(resultArray[11]); //Prints answer as ASCII
Serial.print(resultArray[12]); //Prints answer as ASCII
Serial.print(resultArray[13]); //Prints answer as ASCII
Serial.print(resultArray[14]); //Prints answer as ASCII
Serial.print(resultArray[15]); //Prints answer as ASCII
Serial.print(resultArray[16]); //Prints answer as ASCII
Serial.print(resultArray[17]); //Prints answer as ASCII
Serial.print(resultArray[18]); //Prints answer as ASCII
Serial.print(resultArray[19]); //Prints answer as ASCII
delay(500); //Tells the chip to wait for a half-second
}
```