Goertzel Algorithm Implementation in SYS/BIOS

Detecting both one frequency and all frequencies relating to digits of DTMF (Dual Tone Multi-frequency)

Tireni Omigbodun
Electrical and Electronic Engineering
University of Bristol
Bristol, United Kingdom
ye20178@bristol.ac.uk

Richard Ninan Roy
Electrical and Electronic Engineering
University of Bristol
Bristol, United Kingdom
yi20755@bristol.ac.uk

Neymat Seyidli
Engineering Mathematics
University of Bristol
Bristol, United Kingdom
wq20750@bristol.ac.uk

***Abstract*—This paper illustrates two programs utilising the Goertzel Algorithm, a technique in Digital Signal Processing that can produce faster results over other techniques such as Fast and Discrete Fourier Transform. The first program calculates, using this algorithm, the Goertzel value or relative magnitude from a basic single frequency based sample wave. The other simulates how the Goertzel algorithm would work in an embedded system using dual Tone Multi-Frequency signalling, a method of converting a converting a digit to two frequencies, a large and a small.**

The application of this method is widely seen in telecommunications such as a signalling through a telephone number pad and as already mentioned allows for much faster efficiency rates to be achieved compared to other methods. However, the applications of this method are limited, as only a small number of frequencies can be practically used without performance degradation and by extension, this method can only support the conversion of a small set of digits, for example in our case 16.

***Index Terms*—Goertzel Algorithm, DTMF, SYS/BIOS, TMS320C6000^{TM}**

A. Abbreviations and Acronyms

CSS - Code Composer Studio
IDE - Integrated Development Environment
DTMF - Dual Tone Multi-Frequency.
DSP - Digital Signal Processor
IIR - Infinite Impulse Response filters.
DFT - Discrete Fourier Transform
FFT- Fast Fourier Transform

I. Introduction

The Goertzel Algorithm is a method of tone detection often used in embedded systems. They are used in place of more traditional methods such as FFTs and DFTs. This due to the algorithm being more efficient for the DSPs to process and it uses the required frequencies necessary for detection [1].

For this project, the Goertzel algorithm is used the detection of two tones for DTMF application. The program has three constituent parts:

1) The Identification and Generation of the Input: This cleans the users input and converts it into a signal it composes of DTMF tones.
2) The Goertzel Algorithm Implementation : This takes in the inputs and detects which tones are present in the input.
3) Displaying of the Goertzel Algorithm Output and Detected Digit.

This program is built and debugged on DSP using CCS IDE. The CCS IDE is the platform in which the code for both the generation and detection of the digit are implemented. SYS/BIOS is the operating system which acts as an interface between the CCS IDE and the cores of the DSP being used. It enables real time implementation of the processor used. The specific DSP used in this implementation is the TMS320C66x cores [3]. It has eight cores of which one is used alone for the implementation.

II. Dual Tone Multi-frequency (DTMF)

The DTMF systems are in a form of matrix where the each digit corresponds to two frequency tones. Figure II illustrates the DTMF Table, which has the low and high frequency components and also the digits that are needed to be detected. To ensure the effectiveness DTMF program, the digits are

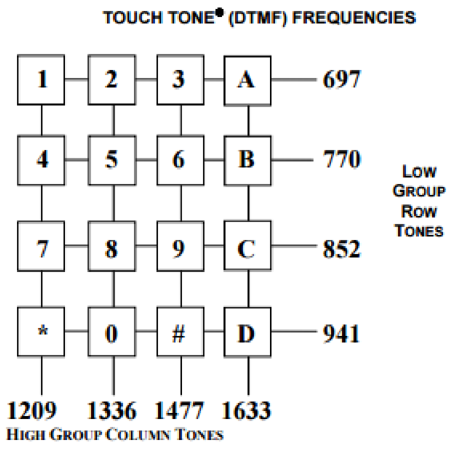
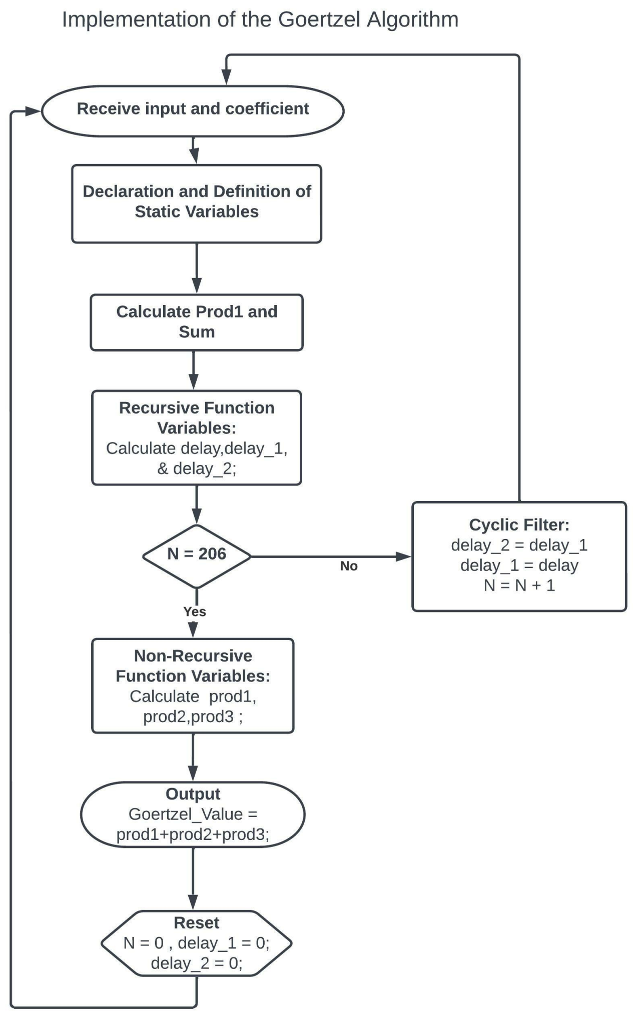


Fig. 1. The DTMF TABLE showing the Digits and their respective Frequencies.

encoded with their respective frequencies and later decoded which involves the use of the Goertzel Algorithm.

III. The Goertzel Algorithm

The Goertzel algorithm is implemented using an IIR filter. The IIR filter is used for implementation instead on the FIR because it uses fewer coefficients and also uses recursive function, which improves efficiency [4]. It uses a Direct form II Canonical IIR Filter Structure. The Goertzel algorithm is a form of DFT filter but for only specific frequencies. The algorithm can be implemented using the following parameters and equations according to [3]:



Implementation of the Goertzel Algorithm
Receive input and coefficient
Declaration and Definition of Static Variables
Calculate Prod1 and Sum
Recursive Function Variables: Calculate delay,delay_1, & delay_2;
N = 206
No
Yes
Cyclic Filter: delay_2 = delay_1 delay_1 = delay N = N + 1
Non-Recursive Function Variables: Calculate prod1, prod2,prod3 ;
Output Goertzel_Value = prod1+prod2+prod3;
Reset N = 0 , delay_1 = 0; delay_2 = 0;

Fig. 2. The Goertzel Algorithms Flowchart

1) IIR Filter Structure: This is the tool necessary for the computation of the algorithm and the definition of the Equations. Its Structure is implemented using a Two-Pole Resonator [3].
2) Recursive Equation: This computes the cyclic nature of the code where it runs a specific number of times.

$$Q_n = input + 2cos(2\pi k/N)Q_{n-1} - Q_{n-2} \quad (1)$$

3) Non-Recursive Equation: This is the equation that calculates the magnitude of the Goertzel output.

$$|y_k(N)|^2 = Q^2(N) + Q^2(N-1) \\ - 2cos(2\pi k/N)Q(N)Q(N-1) \quad (2)$$

4) Integer Frequency k: This is frequency specific and uses the k values to calculate each frequency coefficient.
5) Length of Filter N: 205. Implying that the number of samples that can be implemented by the algorithm.
6) Sampling Rate f_s: 8000Hz. Determines the rate at which data is sampled by the filter.
7) Coefficients θ_k: This is the coefficients $Q(N)Q(N-1)$ in both equations 1 and 2.That is $2cos(2\pi k/N)$. This is implemented in Q15 format because floating point arithmetic cannot be used for IIR filter applications. List of coefficients can be found in [3].

The Algorithms implementation can be modelled in a flowchart as seen in 2. The inputs and the coefficients are received and variables are instantiated. The Recursive implementation begins involving the use of variable **prod1** and **sum**.

$$prod1 = (coefficient \times Q(N-1)) >> 14 \quad (3)$$

Equations 3 and 4 are used in the implementation of 1. The recursive function is implemented and the program checks the number of sample checked is equal to the length of the filter N. If not, the program is looped till N = 206. If so, it runs the Non-recursive function to find the magnitude $|y_k(N)|^2$.. The Non-recursive function can be implemented using variables **prod1**,**prod2** and **prod3**.

$$prod1 = Q(N-1)^2; \quad (4)$$

$$prod2 = Q(N-2)^2; \quad (5)$$

$$prod3 = (coefficient \times Q(N-1)) >> 14 \times Q(N-2) \quad (6)$$

The declared variables are reset and prepares to take in another input and coefficient.
NOTE: For any calculation involving the coefficients, they need to be shifted in order to prevent overflow. The coefficients are scaled by 14 in the code. The reason is that in converting the coefficients to Q15 format, the first six decimal coefficients were divided by 2 for easy implementation.

A. Implementation for One Frequency on CCS

The implementation for frequency 697Hz can be done using Tasks and Clocks. TASKs and Clocks are part of the real time operating system of the SYS/BIOS. Using the Flowchart in Figure 3,The Goertzel cN be implemented by generating clock parameters and triggers the DTMF generator clock. The clock produces an output (sample) which is synchronous to the sampling rate. This is the DTMF input which will be implemented for the Goertzel algorithm.

The second task is ran and it triggers the second clock for detection. The Goertzel Algorithm is implemented and the magnitude is read. The Output seen from running the program for a single frequency 697Hz can be seen in 4.

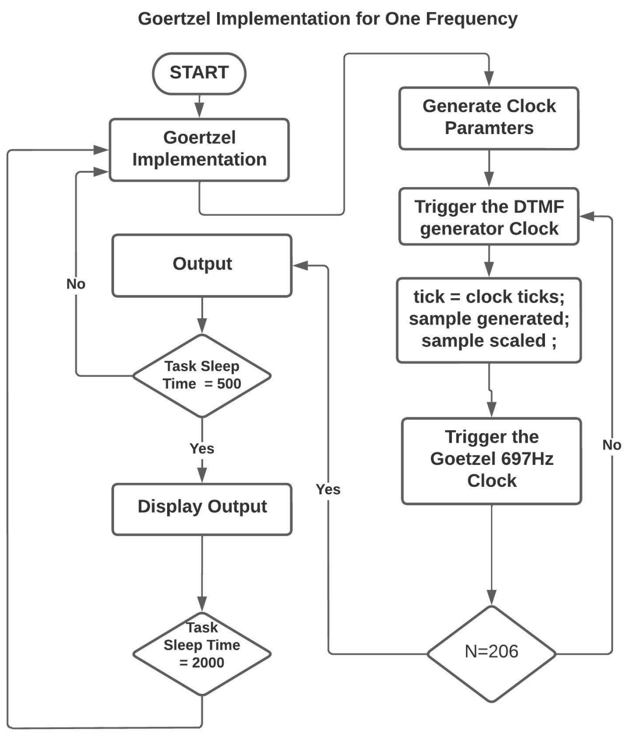
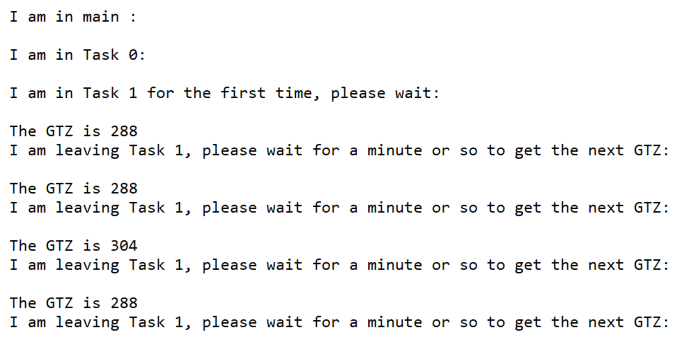

Goertzel Implementation for One Frequency
START
Goertzel Implementation
Generate Clock Paramters
Trigger the DTMF generator Clock
Output
tick = clock ticks; sample generated; sample scaled ;
No
Task Sleep Time = 500
Yes
Trigger the Goetzel 697Hz Clock
No
Yes
Display Output
Task Sleep Time = 2000
N=206

Fig. 3. The flowchart for the frequency 697Hz Goertzel Implementation



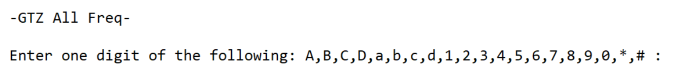
```
I am in main :

I am in Task 0:

I am in Task 1 for the first time, please wait:

The GTZ is 288
I am leaving Task 1, please wait for a minute or so to get the next GTZ:

The GTZ is 288
I am leaving Task 1, please wait for a minute or so to get the next GTZ:

The GTZ is 304
I am leaving Task 1, please wait for a minute or so to get the next GTZ:

The GTZ is 288
I am leaving Task 1, please wait for a minute or so to get the next GTZ:
```

Fig. 4. The Output of the Goertzel Algorithm for One Frequency 697Hz

IV. Detecting digits from keypad

This program first starts with a request to enter a digit given in a list as so:

A. Identification



```
-GTZ All Freq-

Enter one digit of the following: A,B,C,D,a,b,c,d,1,2,3,4,5,6,7,8,9,0,*,# :
```

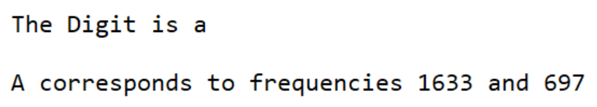
Once entered, the program checks to see if the user input is one of the four lower case letters presented and converts it into an uppercase if that is the case by manually checking it against every lowercase letter possible using an if statement. If there is a match, the user input will be converted to an uppercase letter; DTMF signalling only allows one case of letters to be processed.

Once the user input has been cleaned, it can be compared to an array of DTMF digits in a for loop, comparing the digit to the next member of the array in every iteration. The number of iterations is recorded with n. If there is a match, two integers are calculated.

$$y = \left\lfloor \frac{n}{4} \right\rfloor \tag{7}$$

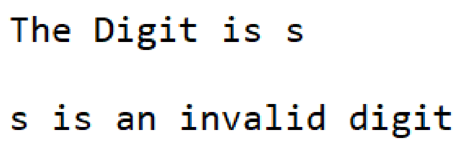
$$x = n - 4y \tag{8}$$

There are two arrays, *Xf* consisting of the high frequencies of DTMF and *Yf* consisting of the low. x and y indicate the index of the element needed in each array respectively. Freq1 is assigned the high frequency components by extracting element x from array *Xf* and similarly, Freq2 is assigned the low frequency components by extracting element y from array *Yf*. Now that the program has confirmed that the user input has yielded two component frequencies, the two magnitudes variables are made equal to 32768. The code also breaks so that the identification is completed and avoids a slower processing time. Finally, the two frequencies relating to the user input are printed in the console:



```
The Digit is a

A corresponds to frequencies 1633 and 697
```

If a digit not from the DTMF array is inputted, the following message will be printed:



```
The Digit is s

s is an invalid digit
```

If a successful match is found for the digit, a variable will be given a new value and if this new value is not detected at the end of the loop, the previous image is displayed.

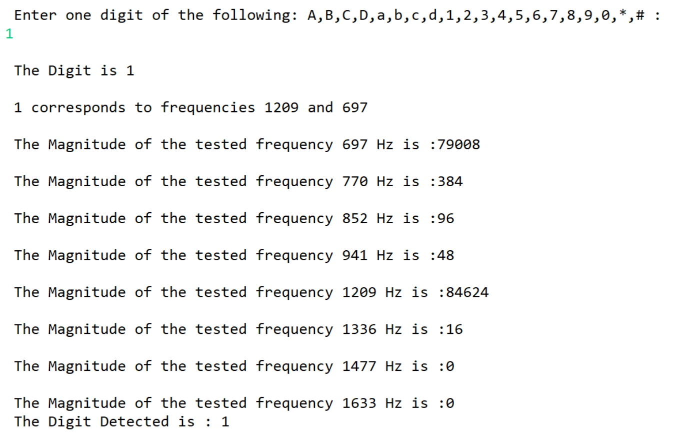
B. Goertzel Algorithm for All Frequencies

The implementation for all frequencies requires the use of two clocks and a Task. The task is triggers the Identification of Digits and sends the frequencies to the DTMF generator. On generating the input by the first clock, the second clock is triggered and checks for the Goertzel value for each frequency for one tick of the clock. Unlike the implementation for one frequency, the algorithm uses an array of 8 for each parameter such as **prod1**, **prod2**, **N**, **sum** and **prod3** to ensure that each frequency is checked at the right tick. The code in 2 is looped 8 times for one tick of the clock.

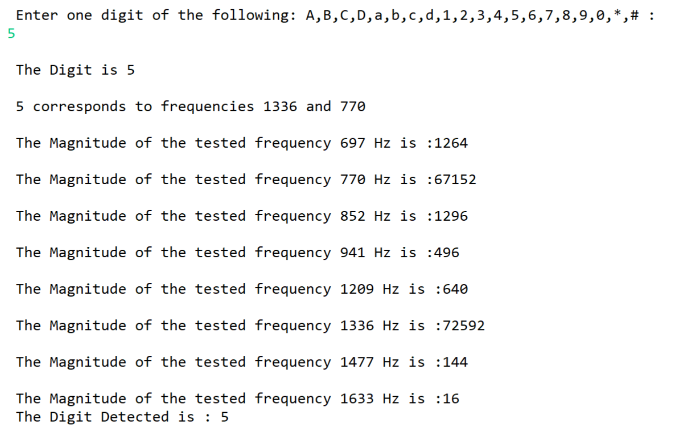
C. Displaying of Outputs

Displaying of Outputs checks the first four elements of the relative magnitude array to find the biggest magnitude (small frequency) as well as doing the same for the last four (large frequency). To check, an iterative process is set up where the nth element of the relative magnitude array is check with the current stored biggest magnitude. If the element is bigger than the current magnitude, that element becomes the biggest magnitude and the current number of iterations becomes the new index number. Once all the elements of the array have been checked, the index numbers are fed into a DTMF matrix array to produce the corresponding digit.

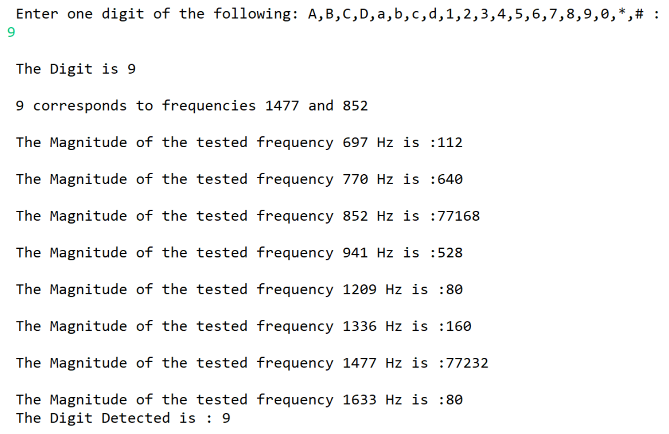
The diagrams below shows the Goertzel Algorithm implementation for numbers 1,5,9 and letter d. It shows the magnitude of each frequency of the DTMF input as well as detecting the correct digit.



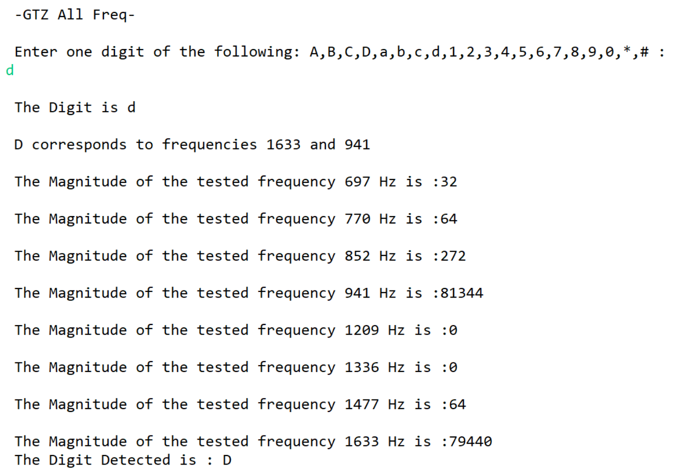
```
 Enter one digit of the following: A,B,C,D,a,b,c,d,1,2,3,4,5,6,7,8,9,0,*,# :
1

 The Digit is 1

 1 corresponds to frequencies 1209 and 697

 The Magnitude of the tested frequency 697 Hz is :79008

 The Magnitude of the tested frequency 770 Hz is :384

 The Magnitude of the tested frequency 852 Hz is :96

 The Magnitude of the tested frequency 941 Hz is :48

 The Magnitude of the tested frequency 1209 Hz is :84624

 The Magnitude of the tested frequency 1336 Hz is :16

 The Magnitude of the tested frequency 1477 Hz is :0

 The Magnitude of the tested frequency 1633 Hz is :0
 The Digit Detected is : 1
```



```
 Enter one digit of the following: A,B,C,D,a,b,c,d,1,2,3,4,5,6,7,8,9,0,*,# :
5

 The Digit is 5

 5 corresponds to frequencies 1336 and 770

 The Magnitude of the tested frequency 697 Hz is :1264

 The Magnitude of the tested frequency 770 Hz is :67152

 The Magnitude of the tested frequency 852 Hz is :1296

 The Magnitude of the tested frequency 941 Hz is :496

 The Magnitude of the tested frequency 1209 Hz is :640

 The Magnitude of the tested frequency 1336 Hz is :72592

 The Magnitude of the tested frequency 1477 Hz is :144

 The Magnitude of the tested frequency 1633 Hz is :16
 The Digit Detected is : 5
```

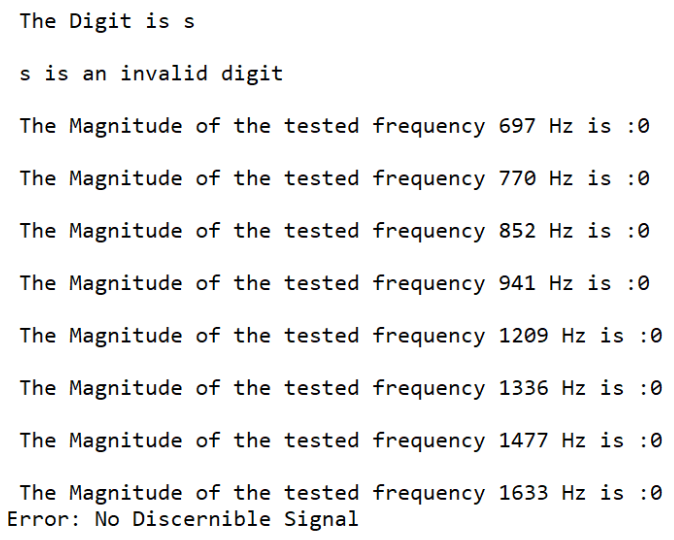


```
 Enter one digit of the following: A,B,C,D,a,b,c,d,1,2,3,4,5,6,7,8,9,0,*,# :
9

 The Digit is 9

 9 corresponds to frequencies 1477 and 852

 The Magnitude of the tested frequency 697 Hz is :112

 The Magnitude of the tested frequency 770 Hz is :640

 The Magnitude of the tested frequency 852 Hz is :77168

 The Magnitude of the tested frequency 941 Hz is :528

 The Magnitude of the tested frequency 1209 Hz is :80

 The Magnitude of the tested frequency 1336 Hz is :160

 The Magnitude of the tested frequency 1477 Hz is :77232

 The Magnitude of the tested frequency 1633 Hz is :80
 The Digit Detected is : 9
```



```
 -GTZ All Freq-

 Enter one digit of the following: A,B,C,D,a,b,c,d,1,2,3,4,5,6,7,8,9,0,*,# :
d

 The Digit is d

 D corresponds to frequencies 1633 and 941

 The Magnitude of the tested frequency 697 Hz is :32

 The Magnitude of the tested frequency 770 Hz is :64

 The Magnitude of the tested frequency 852 Hz is :272

 The Magnitude of the tested frequency 941 Hz is :81344

 The Magnitude of the tested frequency 1209 Hz is :0

 The Magnitude of the tested frequency 1336 Hz is :0

 The Magnitude of the tested frequency 1477 Hz is :64

 The Magnitude of the tested frequency 1633 Hz is :79440
 The Digit Detected is : D
```

If the digits entered deviate from the digits prompted for in the display, then all the relative magnitudes in the result will end up as zero. An error message would show up that the digit entered is invalid. Another error message would say that there is no discernible signal.



```
 The Digit is s

 s is an invalid digit

 The Magnitude of the tested frequency 697 Hz is :0

 The Magnitude of the tested frequency 770 Hz is :0

 The Magnitude of the tested frequency 852 Hz is :0

 The Magnitude of the tested frequency 941 Hz is :0

 The Magnitude of the tested frequency 1209 Hz is :0

 The Magnitude of the tested frequency 1336 Hz is :0

 The Magnitude of the tested frequency 1477 Hz is :0

 The Magnitude of the tested frequency 1633 Hz is :0
Error: No Discernible Signal
```

V. Debugging and Improvements to the code

A. Debugging

Implementing the code resulted in multiple errors during the development period which were resolved and accounted for in the following :

1) Output stabilisation when detecting a specific frequency was brought under control.
2) The Goertzel value output was not accurate for multiple frequencies when detecting a digit from the keypad. The code was altered to correct this error.
3) The coefficients provided had to be used differently depending on the one used, which lead to problems in getting the shift of products and sums right but was eventually manually adjusted for.
4) The programming language (C) used here unlike others cannot be used to detect errors dynamically which lead to late detection of output errors and loss of time.

B. Possible improvements to the code

The implemented code was designed to only be interested in detecting the magnitude of a signal to give an output. One may code for additional improvements for much greater versatility:

1) Addition of a phase detecting code in the algorithm would increase the accuracy of the output.
2) The presence of a fixed time delay in the code to allow for the Goertzel Algorithm calculation time decreases the processing efficiency of each run. A method of communicating between functions to allow one to continue when the other has finished would resolve this.
3) The Coding and Decoding frequency sections do not share commonality in their pointers as a result of inflexible delegation. Commonality would increase the efficiency of the program and make it take less memory.
4) Simple quality of life improvement from a user standpoint could be introduced if so desired. For instance, there is no way to terminate the programs from the console. A simple solution would be to give the user the option to exit the program through an input to the console.
5) The function for user input is `gets()`. This function has no buffer limit to the input and as a result is susceptible to buffer overflow. Another function such as `fgets()` allows for a buffer limit and therefore, is a more reliable method of user input.

VI. Conclusion

The Modified Goertzel algorithm used through DTMF employs significantly less computational power when used to detect a small number of spectrum points than a FFT. But it increases as the same as a DFT when used to detect all of the spectrum range. The authors of the paper try to demonstrate how to implement a program to detect a single frequency in a signal and also to detect the key pressed on a keyboard through its associated frequencies. This shows usefulness of using only signal magnitudes to detect certain frequencies and how computationally easier and efficient it is when used for a small number of spectrum points.
Further work can be carried out by improving the code efficiency and optimising the code to detect phase of the signals for increased output accuracy.

A. Applications of Goertzel Algorithm

Some applications that employ the Goertzel Algorithm are discussed below:

1) The major application of Goertzel algorithm is in the recognition of DTMF tones produced by the push buttons of the keypad of a traditional analogue telephone.
2) It can be used in reverse as a sinusoid synthesis function, which requires only one multiplication and subtraction per generated sample
3) In embedded systems where direct measurement of physical processes are taken the value of real life data is precious. Hence these sections where only the magnitude is the only part we are concerned with we can employ this algorithm to extract our real-valued data.

References

[1] Staff, E., 2022. The Goertzel Algorithm - Embedded.com. [online] Embedded.com. Available at: https://www.embedded.com/the-goertzel-algorithm/ [Accessed 5 June 2022].

[2] Y. Huang, J. Chen and Q. Wang, ”Encoding and Decoding of DTMF Signal Based on DSP,” 2018 2nd IEEE Advanced Information Management,Communicates,Electronic and Automation Control Conference (IMCEC), 2018, pp. 478-481, doi: 10.1109/IMCEC.2018.8469726.

[3] N. Dahnoun, Digital Signal Processing Implementation: Using the TMS320C6000 DSP Platform, Prentice Hall, 2000.

[4] N. Dahnoun, ”Multicore DSP: From Algorithms to Real-time Implementation on the TMS320C66x SoC,” 2018.

[5] Elmenreich, Wilfried (August 25, 2011). ”Efficiently detecting a frequency using a Goertzel filter”