Digital Implementation of Signal Generator

Patel Ashish
The Maharaja Sayajirao University of Baroda

Abstract

Function generators are used as signal source in development, test and repair of electronic equipment and circuits in laboratory. These function-generators use active and passive components to generate waveforms like sine wave, pulse wave, triangular wave and ramp waveform. So, limited numbers of waveform can be generated. Sometimes we need to generate complex shaped waveform for testing purpose e.g. 3^{rd}, 5^{th} harmonics, exponential function etc. These functions can't be realized with function generators available in laboratory. So, function generator is designed with microcontroller where the equation of function can be stored and calculated. These arbitrary waveform generators, or AWGs, are sophisticated signal generators which allow the user to generate arbitrary waveforms, within published limits of frequency range, accuracy, and output level. Unlike function generators, which are limited to a simple set of waveforms; an AWG allows the user to specify a source waveform in a variety of different ways.

1. Introduction

Signal generator is designed and implemented to generate waveform of any shape which can be described in its mathematical equation. To generate periodic waveform of given equation, frequency and amplitude are required. The amplitude can be change easily by the gain of the amplifier. But, changing the frequency is not as simple as amplitude with microcontroller.

Frequency is very important parameter for any waveform. To generate waveform with microcontroller and DAC, two types of timing have been considered. One is the time period of the waveform and another is the sampling time (Nyquist criterion).

Atmega32 microcontroller is used for implementation purpose because of its RISC architecture which facilitates single clock cycle execution.

As the function generator is implemented in the microcontroller, a buffer is required in the microcontroller to hold the sample values. Samples are calculated for the whole cycle and are stored in the buffer i.e. in the RAM of the microcontroller. From this buffer, data is fetched and transferred to the DAC port and pointer to the next sample data is updated every time. After completion of one cycle, i.e. one time period, and pointer is updated to the start of the buffer and this process is repeated again. Also very importantly, the timing calculation is done by the microcontroller before storing samples in the buffer to generate the waveform of desired frequency. Interrupt is generated from user interface (keys) to change the frequency and/or shape of the waveform. When interrupt is arrived to generate different waveform then, buffer is filled with the samples values calculated for new frequency and/or new shape.

Sampling time, Ts, is the optimal period between two samples. Since this signal generator is implemented in the microcontroller, time is required to take data from the buffer where the samples are stored, and then output the data to the DAC port, and digital to analog conversion. DAC was selected in such a way that its conversion time is less than the sampling time.

2. Design Consideration

Let, f is the frequency of the waveform that is to be generated. Thus, Numbers of samples required

$$N = \frac{1}{Ts * f} \quad \ldots\ldots\ldots\ldots\ldots 1$$

In microcontroller memory is of limited size. So the buffer which is designed to store the sample values of the waveform has some maximum capacity, BUFFER_MAX_SIZE, and its size is not infinite. For example consider the frequency of the waveform is 2 Hz and the sampling time of the microcontroller is 2us. So the samples required are 250000 according to the

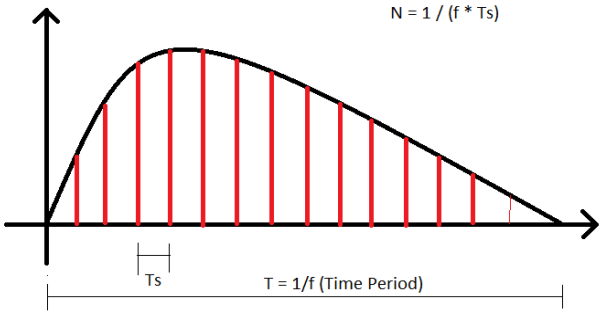

Figure 1: Calculating numbers of samples

above formula of N and therefore 250kB of memory is required in the microcontroller, which may not be available. For atmega32, which is used to design this signal generator has maximum memory of 2kB (SRAM). Also, if frequency further decreases then the required samples and hence the size of the buffer required will increase. So theoretically we need and microcontroller with much large memory to generate the waveform of very small frequency.

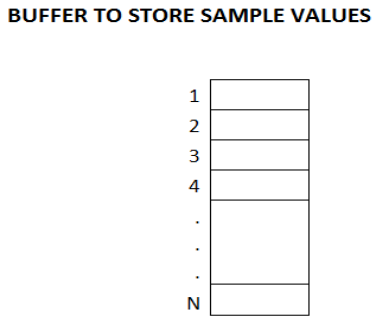

Figure 2: RAM storage in microcontroller

Another practical problem in implementing this signal generator occurs at the high frequency. As microcontroller and DAC are used to design the signal generator, there is the finite delay between two samples. In microcontroller it cannot be made to zero because of finite time required to fetch the sample from buffer to port and DAC conversion time. So the sampling frequency has maximum upper limit. In above example sampling time is assumed to be of 5us. So the Nyquist sampling frequency is 200 kHz. So from Nyquist criterion we cannot generate waveform whose frequency is above 100 kHz. Also, from the above equation 1, it is seen that as the frequency of waveform increase the samples N is decreases. But samples are not allowed to decreases so much so that the output is distorted. So sampling time Ts must be decreased to sustain sufficient samples for high frequency waveform.

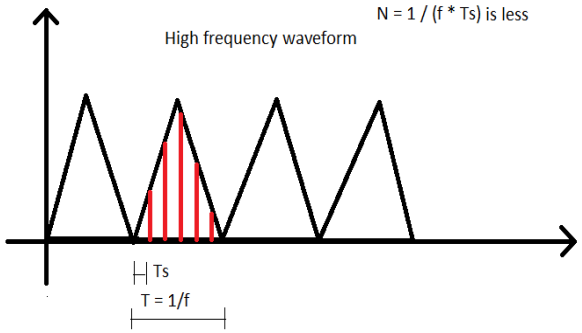

Figure 3: Sampling of high frequency waveform

3. PRACTICAL ISSUES IN GENERATING WAVEFORM WITH MICROCONTROLLER:

3.1 LIMITED BUFFER SIZE (LOW FREQUENCY)

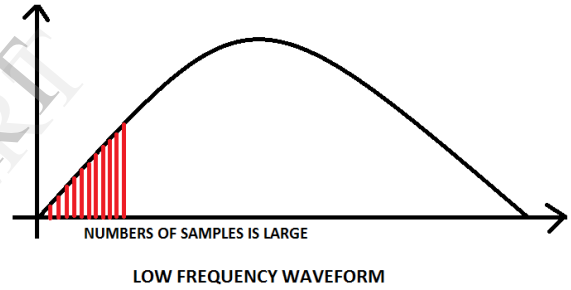

Figure 4: Large numbers of samples

In microcontroller, the memory is of limited size. So the buffer to hold the samples is also of limited size. Maximum buffer size that can be allotted in microcontroller is assumed to be BUFFER_MAX_SIZE in further discussion. Low frequency implies greater time period. So N might become very large to overflow the buffer. So maximum time period of the waveform can be [MAX_BUFFER_SIZE * Ts (Sampling Time)] .Here frequency will be minimum. So, limited size of buffer limits the minimum frequency that can be generated. To overcome this problem delay between two samples is added deliberately, i.e. the sample time is increased. In equation 1, sample time T_s is replaced by (T_s + nT_s), where n is an integer value. So the new sampling time T_e = $(1+n)T_s$.

$$\text{NumberOfSamples} = \frac{1}{f * Ts * (1+n)} \ldots\ldots 2$$

Here, the sampling time is increased by the factor of (1+n). Now, how the best value of n for given

frequency is decided? For that, firstly it is number of the samples is assumed to be the size of available buffer, i.e. BUFFER_MAX_SIZE. It is assumed so to utilize the whole buffer and produce maximum numbers of samples. This BUFFER_MAX_SIZE is put in above equation and we solve that equation for n.

$$n = \frac{1}{f * Ts * BUFFER\_MAX\_SIZE} - 1 \dots \dots 3$$

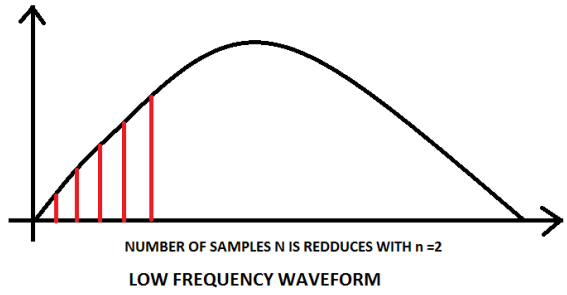

Figure 5: Reduced number of samples

In case of rational number, n is rounded to the next integer number. It must be done because the register in the microcontroller is used to hold this count value n. And only integer values can be stored in it. So n must be rounded to the integer. Also n must be rounded to the next large integer. For example if n turns out to be 2.42 then it is rounded to 3 or greater than 3, but not 2. Now using this calculated value of n, equation (2) with sampling time $(1+n)T_s$ is solved for the number of samples N, and actual buffer size required is obtained, which is less than or equal to BUFFER_MAX_SIZE, but still it is closer to BUFFER_MAX_SIZE. So that, the available buffer efficiently is used. In above example, if n is rounded to 2 then number of samples increases the BUFFER_MAX_SIZE.

For example, consider microcontroller clock frequency is 16 MHz, sampling time T_s to be 10us, BUFFER_MAX_SIZE to be 500, and the frequency is generated is 2 Hz. Now using equation 1, if number of samples required is calculated, which turns out to be 50,000.

$$N = \frac{1}{2 * 10^{-5}} = 50{,}000$$

And maximum buffer size is 500, which is 100 times less than the required samples. So, the sampling time must be increased to decrease the number of samples. Equation 3 is used to calculate the integer value of n with number of samples equals to the BUFFRE_MAX_SIZE, which turns out to be 99.

$$n = \left(\frac{1}{2 * 10^{-5} * 500}\right) - 1 = 99$$

Using this value of n, no of samples required is again calculated, which is 500, i.e. BUFFER_MAX_SIZE.

$$N = \frac{1}{2 * 10^{-5} * (1 + 99)} = 500$$

So the size of the buffer is efficiently used. If n is not integer than it's rounded up value is used to calculate the total samples required, which is less than BUFFER_MAX_SIZE, but never greater than BUFFER_MAX_SIZE. Also, n greater than 99 can be used in the above example and in that case also the required buffer size turns out to be less than BUFFER_MAX_SIZE. But it reduces the number of samples. So to increases the low frequency range of the digital signal generator, microcontroller with large RAM must be selected.

So, low frequency limit can be expanded by,

1. Using the microcontroller which has greater RAM memory.
2. Increasing sampling time interval between two samples.

3.2 NYQUIST CRITERION (HIGH FREQUENCY)

Sampling time T_s includes the time to take data from buffer, output that data to DAC port and update the index of the buffer. So there are finite numbers of instructions that must be executed between two samples. Hence, T_s has some finite value. As the frequency that is to be generated is increased, then from equation 1, number of samples is decreases.

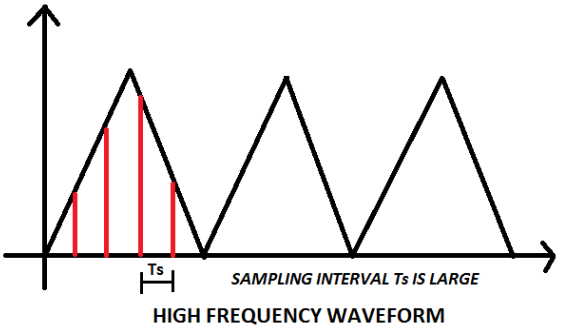

Figure 6: Small numbers of samples

To maintain the sufficient samples N, sampling time must be decreased. But, T_s is the time required to execute the instructions between two samples. So how T_s can be decreased? One way to decreases

the sampling time is to optimize the code which is used to generate the waveform, i.e. the code which is written to take data from buffer and output that data to the DAC port with minimum numbers of instructions. So code must be highly optimized. Another way to decreases the sampling time is to increases the system clock frequency. As the crystal with high frequency is used then the execution time of the instruction decreases, which in turn decreases the sampling time.

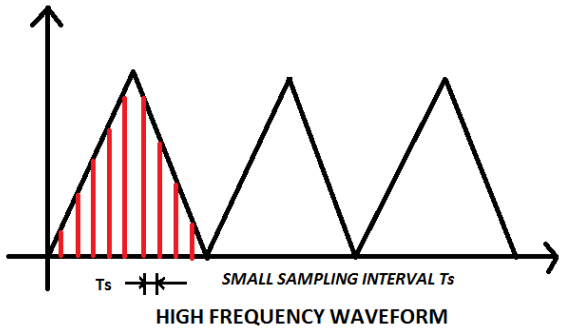

Figure 7: Increased numbers of samples

To calculate the numbers of samples for high frequencies, equation 1 is used. For high frequencies, N turns out to be less than BUFFER_MAX_SIZE. So there isn't any problem regarding size of the memory of the microcontroller.

$$Ts = \frac{No\_Of\_Instruction\_To\_be\_Executed}{Clock\_Frequency}$$

For example, consider microcontroller with clock frequency of 16 MHz, number of instructions that must be executed between two samples is 80 and frequency that is to be generated is 20 kHz. Here RISC microcontroller is used to implement signal generator. So every instruction executes in one clock cycle. Now, the time to executes 80 instructions between two samples is (80/16000000) = 5us. So the sampling time is 5us. Using calculated T_s, numbers of samples N can be calculated using equation 1, which turns out to be 10 and it is much smaller than the available buffer size. So T_s must be decreases to increase N. In this example code must be optimized i.e. number of instructions that are executed between two samples must be reduced because T_s decreases with decrease in the instructions. Crystal frequency can also be increased because T_s decreases with increase in the crystal frequency. We cannot decrease Ts as much as we required, because DAC has some finite conversation time, which is of the order of us to few hundreds' of ns.

T_s can be minimized by,

1. Optimizing the code that is executed between two samples (Generally data movement code).
2. Using the crystal with higher frequency to reduce the instruction execution time.

4. SYSTEM IMPLEMENTATION

Algorithm to fill the buffer with samples calculated from the equation of function f(x) and frequency of the waveform. This algorithm is executed every time when new function for wave shape and/or new frequency to be generated is entered by user.

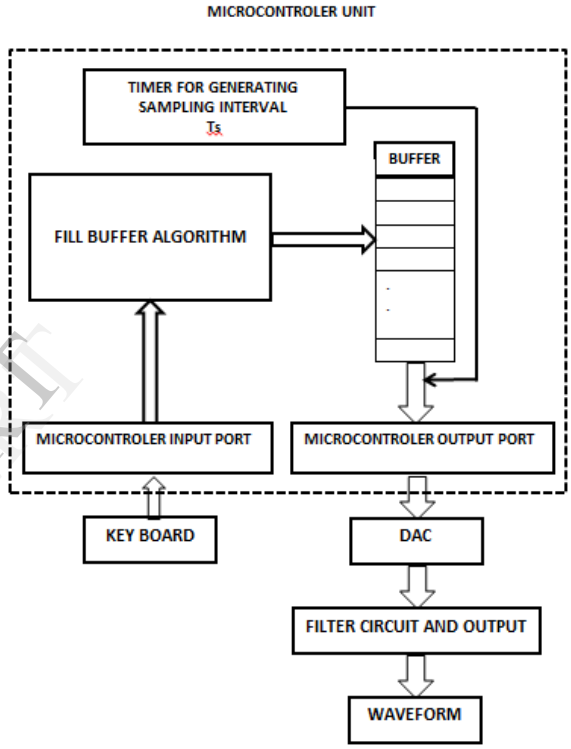

Figure 8: System Layout

Fill Buffer algorithm

```
Fill Buffer (Frequency, f(x)) {
//Calculate number of samples
```

$$N = \frac{1}{(f * Ts)}$$

```
//Decide which equation to use (Low frequency or
//High frequency equation) ……
If (Numbers Of Samples <=
BUFFER_MAX_SIZE) {
//High frequency
Register Count = 0;
        n = 0;
} else {
//Low frequency
```

$$n = \frac{1}{f * Ts * BUFFER\_MAX} - 1$$

$$N = \frac{1}{f * Ts * (1 + n)}$$

Register Count = (int) n;
}
//Fill the buffer with sample values
For (i = 0; i < Number Of Samples; i ++) {
Buffer[i] = f ($i * \frac{Ts}{N}$);
}
}

Algorithm to output data from the buffer, filled with the sample values, to the DAC port. This block of code executes at every sampling time period Ts. Timer is used for generating sampling time Ts.

Main algorithm

```
Main () {
        While (1) {
//register count is used for n
//After n iteration this loop will be executed
If (Timer completed = = true) {
If (i > = No Of Samples) {
i = 0;
}
DAC_PORT=BUFFER[i];
Timer completed = false;
}
}
}
```

```
Interrupt sub routine For Timer () {
//Timer for time period equals to Ts
If (Current Register Count == 0) {
Current Register Count = register Count (integer value of n) + 1;
Timer completed = true;
} else {
Current Register Count--;
}
}
```

5. CONCLUSION

Implementing signal generator in microcontroller provides great flexibilities to generate complex shaped periodic waveform. It provides adequate control on frequency as well as amplitude. One can also use random function to generate noise for testing purpose.

6. References

[1] Muhammad Ali Mazidi; 8051 microcontroller and embedded systems; 2nd Edition, 2011.
[2] Jacqueline Wilkie, Michael Johnson and Reza Katebi; Embedded system: An Introductory Course; Palgrave Macmillan,2001
[3] Gayakwad. R.A, Op-amps and linear integrated circuits;4th Edition, 2010
[4] Jouko Vankka ; Direct Digital Syaathesizer; 2nd Edition,2001
[5] Richard Barnett, Larry O'Cull and Sarah Cox; Embedded C Programming and the Atmel AVR; Cengage Learning, 2006
[6] Christian Diedrich, Francesco Russo, Ludwig Winkel ,Terry Blevins; Function Geberator Application in RF System Based on IEC 61804;
[7] Proakis; Digital Signal Processing: Principles Algorithms and Implementation; Forth Edition - 2007

Author Profile

Ashish Patel received the B.E. in Electronics Engineering from The Maharaja Sayajirao University of Baroda. He is currently working in the R&D laboratory. His areas of interests are VLSI, Software Programming and Embedded Systems.