

# Frequency Multiplier for Low Frequency Pulse Width Modulation Signal – A Practical Approach

H. Ferdinando<sup>1</sup>, H. Thiehunan<sup>2</sup>, J. S. Putra<sup>1</sup>

<sup>1</sup>Dept. of Electrical Engineering, Petra Christian University, Indonesia

<sup>2</sup>PT FESTO, Indonesia

*Abstract* – using Pulse Width Modulation (PWM) technique in producing variable voltage controlled by digital system is very common. Usually, PWM is operated above certain frequency. Using low frequency PWM produces bad response for DC motor. The DC motor will turn abruptly and look like unstable. This system is built based on the PLC Festo FC34. This PLC can produce PWM signal, max 1kHz. User can vary its duty cycle by unit in low frequency, but for 1kHz, there are only three choices, i.e. 0%, 50% and 100%. The system was built based on AT89C2051, general purposed microcontroller. It uses 12MHz crystal with 8-bit DIP switch to select the multiplier factor. Input PWM signal is from FC34 directly. The system is applied to control the speed of DC motor. The experiments showed that the error is getting bigger for bigger multiplier factor. For multiplier factor equal to 250, the system produced the biggest error. This is due to the inaccuracy of the microcontroller to do the multiplication operation. Another reason is that the computation speed of the AT89C2051 is too slow for this application. According to the experiment with DC motor, the system can work well with the FC34 to control the speed of DC motor.

*Keywords* – frequency multiplier, PWM, PLC, FC34

## I. INTRODUCTION

PWM (Pulse Width Modulation) signal is a common signal in digital world. One can use to generate analog DC voltage controlled by digital system. PWM is a square wave signal with varied duty cycle. This variation gives different DC voltage; it is the average voltage of the signal. Two important factors in PWM signal are frequency and duty cycle.

Using the PWM signal with low frequency is not recommended, especially for plant with high dynamic behavior such as DC motor. To use low frequency PWM signal in DC motor will make it not turn smoothly, but abruptly. To overcome this problem, a frequency multiplier for PWM signal is needed. The system receives PWM signal with certain duty cycle and produce another PWM signal with the same duty cycle but higher frequency.

System was built based on the MCS-51 microcontroller, i.e. AT89C2051 with 12MHz crystal oscillator. The system was applied to control the speed of a DC motor with certain algorithm. The controller is PLC Festo FC34 with Fuzzy Logic driver.

## II. PULSE WIDTH MODULATION

PWM is very popular in control with digital system such as microcontroller and microprocessor. One uses the PWM to generate varied, usually, analog DC voltage. The operation is simple, i.e. user can simply vary the duty cycle of the square wave but the period is constant. Figure 1 shows several PWM signals with several duty cycles.

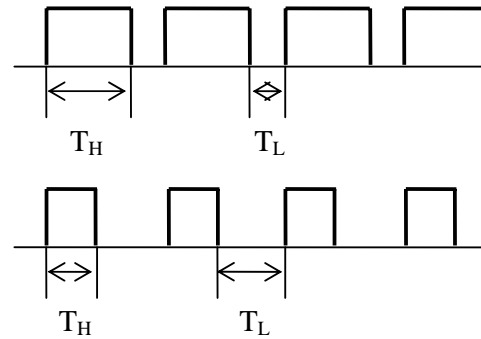


Fig. 1. Several PWM signal

Two important parameters in PWM signal are frequency and duty cycle. One must choose the frequency of the PWM signal before using it, but usually, it is around 1kHz. To use lower frequency is not recommended. With low frequency, the load can 'sense' the HIGH and LOW variation of the signal and it will influence the operation of the whole system. This low frequency is not suitable for plant with high dynamic behavior such as motor. Using low frequency PWM in motor make it rotate abruptly for the motor can 'sense' the HIGH/LOW condition.

The duty cycle has important role during operation, for the frequency is constant. With duty cycle variation, one can have different analog voltage from the same PWM signal. Duty cycle is represented in percent, its value is from 0% to 100%. 0% means the output is 0V while 100% gives maximum output voltage.

## III. PROBLEM FORMULATION

The problems came from the FC34, PLC Festo. This PLC can produce PWM signal without additional code. User only gives the duty cycle to the FC34 and the PLC will generate PWM signal. First, user must select the

frequency of the PWM signal. The FC34 can produce PWM signal from 20Hz to 1kHz [1]. Unfortunately, user does not have freedom in choosing the duty cycle. For 20Hz, the duty cycle's resolution is 1%. It means one can choose any duty cycle, such as 11%, 59%, etc. The higher the frequency, the worse the duty cycle, e.g. for 1kHz, user can only choose among 0%, 50% and 100%.

The problem raised when user wants to have higher frequency PWM signal and good resolution for the duty cycle. This cannot be accomplished in the FC34.

#### IV. FREQUENCY MULTIPLIER

The frequency multiplier in this project is for PWM signal only, i.e. square wave. The system reads the width of HIGH and LOW pulse from input signal. If the multiplier factor is two, both widths are divided by two. For 20Hz PWM signal with duty cycle 50%, both pulses have width 25ms. If the multiplier factor is 5 then the new width for both pulses is 5ms. The period of the new PWM signal is 10ms, it is equal to 100Hz PWM signal.

#### V. HARDWARE DESIGN

The hardware is simply single chip microcontroller AT89C2051 with 12MHz crystal oscillator. The input PWM signal is from the FC34. To see the performance of this frequency multiplier, the system is applied to control the speed of DC motor. There is a DIP switch to select the multiplier factor. It is 8-bit, so there are 256 conditions. Figure 2 shows the block diagram of the system.

The input PWM signal from the FC34 is fixed at 20Hz; therefore, the duty cycle can be varied with 1% resolution. This input signal goes to external interrupt of AT89C2051. The AT89C2051 will read the width of both pulses.

The FC34, like any other PLCs, uses 0-24V to represent LOW and HIGH condition. Unfortunately, the microcontroller uses 0-5V for LOW and HIGH. This needs certain interface, i.e. simple voltage divider.

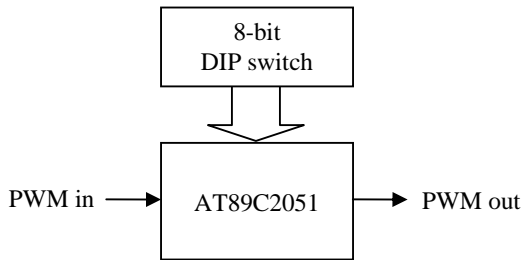


Fig. 2. Block diagram of the system

For the system is applied to DC motor with internal encoder, there is special interface between encoder and PLC. The encoder level voltage is 0-5V. It does not match

with the PLC. To overcome this problem the system uses a transistor. The transistor uses the opto isolator in order to give optical interface to guard the PLC.

#### VI. SOFTWARE DESIGN

The AT89C2051 has two internal timers, Timer0 and Timer1. Timer0 is used to measure the width of HIGH and LOW pulse, while the other is used to generate output signal. With 12MHz crystal oscillator, a cycle in the AT89C2051 is 1 $\mu$ s [2]. This will be used for calculation.

The external interrupt is set to generate interrupt when there is edge transition, HIGH to LOW. To read the width of HIGH and LOW pulse, two external interrupt are used. Transition from LOW to HIGH starts the timer to measure the width of HIGH pulse. This timer will be stopped when there is transition from HIGH to LOW. The similar, but opposite, operation is done to measure the width of the LOW pulse. In order to have HIGH to LOW transition for both external interrupt, one of the external interrupt needs NOT gate.

The detail operation is when there is transition from HIGH to LOW, timer0 (HIGH pulse) is stopped, the values are read and timer0 is started again. The other condition has the same process. The value for each timer is saved in THx and TLx, for the timer is 16-bit. The timer value is needed for multiplier calculation.

The following discussion is about the main operation inside the AT89C2051, i.e. dividing TH0 and TL0 by multiplier factor. It is easier to use a case study to explain the algorithm. The input signal is fixed 20Hz (its period is 50ms) with duty cycle 50%.  $T_H = T_L = 25$ ms. Equation (1) shows how 25ms is represented by 16-bit register THxTLx (e.g. it is timer0)

$$TH_0 TL_0 = \frac{25ms}{1\mu s} = 25000 \quad (1)$$

Value 25000 is distributed in TH0 and TL0, and TH0 is

$$TH_0 = \frac{25000}{256} \cong 97 \quad (2)$$

This value experiences floor rounding. The value of TL0 is little bit complex, i.e.f

$$TL_0 = 25000 - 256 * 97 = 168 \quad (3)$$

The value of TH0 is incremented by one when TL0 overflows, i.e. when there is transition from 255 to 0. This applies to both timer0 and timer1. If TH0 overflows 97 times and TL0 is 168, then the timer reads 25000 or 25ms.

With multiplication factor of 50, this 25000 is divided by 50. The expected result is 500. Here is how to get the new TH0 and TL0:

$$TH'_0 = \frac{TH_0}{50} = \frac{97}{50} = 1 \quad (4)$$

Division in equation (4) has remainder, i.e. 47. The TH0 is 256 times TL0, we cannot neglect this remainder. Next, the division for TL0 is done, see equation (5). It also has remainder.

$$TL'_0 = \left( remainder\_TH_0 \frac{255}{50} \right) + \frac{TL_0}{50} \quad (5)$$

$$TL'_0 = 47 \frac{255}{50} + \frac{168}{50} \cong 235 + 3$$

In the calculation, if 255 is divided by 50, the remainder is 5; therefore there is an accumulation of this remainder 47 times. This also contributes to the new TL0. This value is

$$sisa = \frac{47 * 5}{50} \cong 4 \quad (6)$$

The result in equation (6) still has remainder, i.e. 35. This value will be neglected compare to the multiplier factor. So, the new TL0 will be

$$TL'_0 = 3 + 235 + 4 = 242 \quad (7)$$

The final result of TH0 and TL0 is 1 and 242 respectively. It means TH0 will overflow once and TL0 is 242. As TH0 is 256 times TL0, this combination gives timer value 498 instead of 500 (the expected timer value). These long processes are repeated for TH1 and TL1 for the duty cycle is 50%.

With 498 as the new timer value, the HIGH pulse is 498 $\mu$ s (see equation (8)) instead of 500 $\mu$ s. The resulted signal now has new frequency, i.e. 1004Hz, see equation (9). The expected value is 1kHz. The error came from the rounding during the division process.

$$T_H = 498 * 1\mu s \quad (8)$$

$$f = \frac{1}{(498 + 498)10^{-6}} Hz = 1004Hz \quad (9)$$

The new value of TH0 and TL0 is put into register TH1 and TL1 respectively. This timer is used to generate PWM signal with higher frequency and the same duty cycle.

The controller is FC34, it receives information about the speed of the DC motor via encoder. The pre-experiment shows that the maximum speed is 3600 RPM with 7000 pulse per second. For the PLC's fast counter is limited to 2kHz (2000 pulse per second), pulse from the

encoder must be divided by 4. It makes the number of pulse in a second is 1750, it is equal to 3600 RPM.

Inside the FC34, there is Fuzzy Logic kernel called the FuzzIPC. This will help user to implement fuzzy algorithm in the PLC [3]. The system uses two input, i.e. *error* and *derror*. Error is PV (present value) minus SP (setting point) and derror is difference between current and previous error. Software for FC34 is written using statement list [1]. Programming in statement list looks like programming using assembly language. It is different from the common programming language in the PLC, i.e. ladder diagram. Listing 1 shows some of the statement list in the FC34

LISTING 1  
PLC FESTO'S SOFTWARE

```
STEP 0
IF N T_S 'TimeSampling
THEN LOAD PV 'PV (Rps)
- SP 'SP (Rps)
TO ERROR 'Error (1)

LOAD L_ERROR 'Error (n-1)
- ERROR 'Error (1)
TO dERROR 'Selisih Error

LOAD ERROR 'Error (1)
TO L_ERROR 'Error (n-1)
```

PLC uses a modul called *fastout* to generate PWM signal. The chosen frequency is 20Hz for this frequency can be controlled with duty cycle for any values between 0% and 100%.

For this purpose, there are several functional unit (FU) which must be set [1]. There are 5 FU inputs, i.e. FU32 (0: *init*; 1: *stop* output; 2,3,4,5: *start*), FU33 (*Output number* 0...7), FU34 (*On time* \*0.5mS), FU35 (*Off time* \*0.5mS), FU36 (*Number of pulses*) dan 3 FU outputs which consist of FU 32 (0 *if successful*; 100 *if driver not installed*), FU33 (1 *if output started and not yet finished*), FU34 (*Number of remaining pulse*).

The next part is developed to read the pulse to calculate the speed of the motor. It uses fas counter [1], it couns the number of pulse in 1 second. There are several FUs in this modul, i.e. FU32 (0: *reset*; 1: *parameterizing*; 2: *activating*; 3: *interrogating the status and current counter value*), FU33 (0: *first counter*; 1: *second counter*).

## VII. EXPERIMENTS

The first experiment was for duty cycle 50% with several multiplier factors. It is shown in table 1. The result shows that there is some difference between calculation and implementation. This is due to the rounding as explained before. Another reason for this difference is that this system uses only one timer to measure the width of HIGH and LOW pulse of the PWM wave. When the timer is stopped and its timer value is transferred to certain memory location, several ticks are missing at this time.

TABLE 1  
EXPERIMENT WITH DUTY CYCLE 50% AND  
SEVERAL MULTIPLIER FACTOR

Faktor Pengali	Perhitungan (Hz)	Pengujian (Hz)	Selisih (Hz)
1	20	20.0	0.0
10	200	200.0	0.0
20	400	394.5	-5.5
30	600	588.2	-11.8
40	800	772.2	-27.8
50	1000	972.8	-27.2
60	1200	1157.4	-42.6
70	1400	1362.4	-37.6
80	1600	1538.5	-61.5
90	1800	1712.3	-87.7
100	2000	1912.0	-88.0
110	2200	2083.3	-116.7
120	2400	2252.3	-147.7
130	2600	2500.0	-100.0
140	2800	2611.0	-189.0
150	3000	2809.0	-191.0
160	3200	2941.2	-258.8
170	3400	3125.0	-275.0
180	3600	3300.3	-299.7
190	3800	3484.3	-315.7

Next figures shows the experiments for duty cycle variation from 5% to 95% with several multiplier factors. The calculation (diamond) and the implementation (box) are plotted in the same graph.

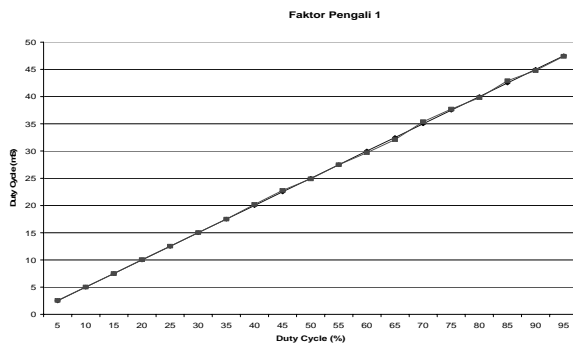


Fig. 3. Multiplier factor = 1

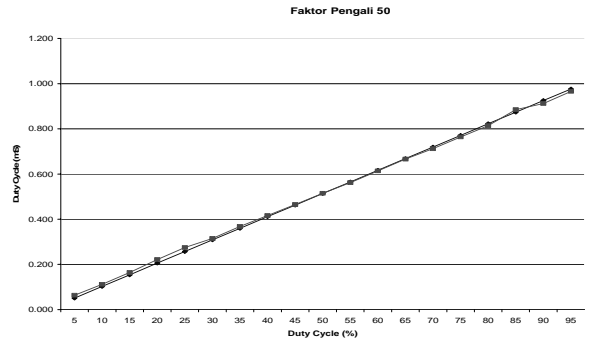


Fig. 4. Multiplier factor = 50

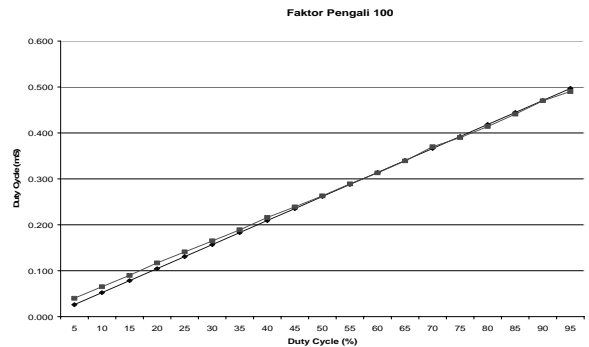


Fig. 5. Multiplier factor = 100

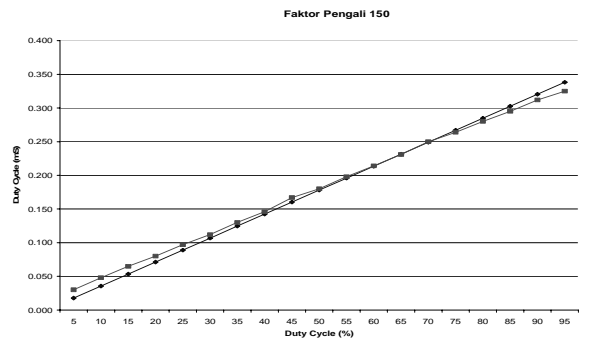


Fig. 6. Multiplier factor = 150

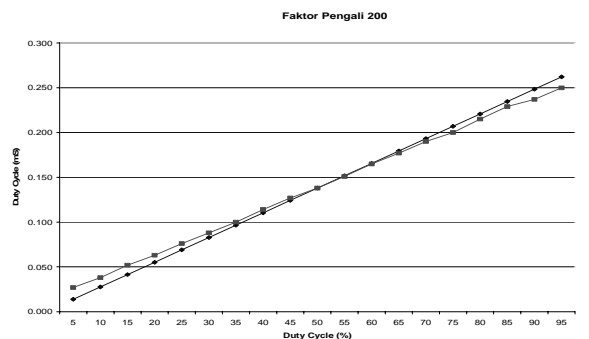


Fig. 7. Multiplier factor = 200

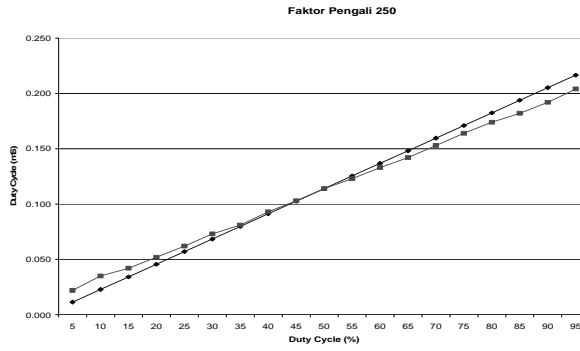


Fig. 8. Multiplier factor = 250

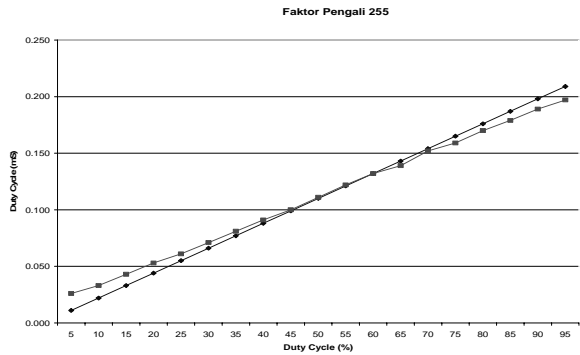


Fig. 9. Multiplier factor = 255

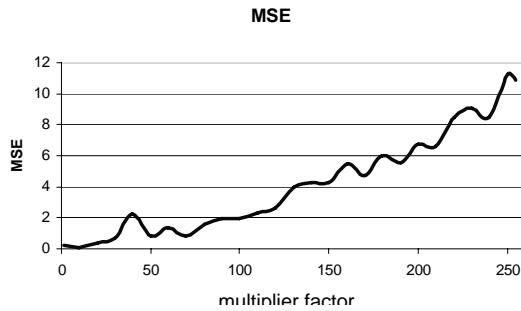


Fig. 10. Performance of all multiplier factor experiments

For all experiments using different multiplier factor, it is shown that the higher the multiplier factor, the larger the error of the system. Using higher multiplier factor make the PWM signal have higher frequency. High frequency signal need faster computation time compare to the low frequency.

Figure 11 shows the response of the motor for due to duty cycle variation. The expected result is that the duty cycle and the speed of the motor has linear response. This figure shows something interesting for using PWM wave from PLC directly and using multiplier factor = 1. Small duty cycle means the HIGH pulse is narrow, this makes the response is not good, the gradient is gradually changing. This explains the explanation about using low frequency PWM signal in motor.

The result shows that the linear response is occurred for higher multiplier factor. The most linear response is for 255. With 20Hz input, using 255 as the multiplier factor gives PWM wave with 5.1kHz. This is quite high for the PWM signal. Unfortunately, this multiplier factor produce large error as shown in figure 10.

This goal of this experiment is to show that the system can be used together with certain control algorithm, in this case fuzzy logic. The whole system can work well although for several points there is some error.

## VIII. DISCUSSION

The division result of PWM signal at 20Hz is not satisfied. The performance of the system can be analyzed with the MSE of the duty cycle, see figure 10. Multiplier factor from 1 to 10 are still acceptable, the MSE is small. This is due to the remainder of the division. The higher the multiplier factor the closer the remainder to that multiplier factor. Take an example of multiplier factor = 200, the maximum remainder is 199. This remainder contributes 199 $\mu$ s or 0.199ms of difference from the expected frequency.

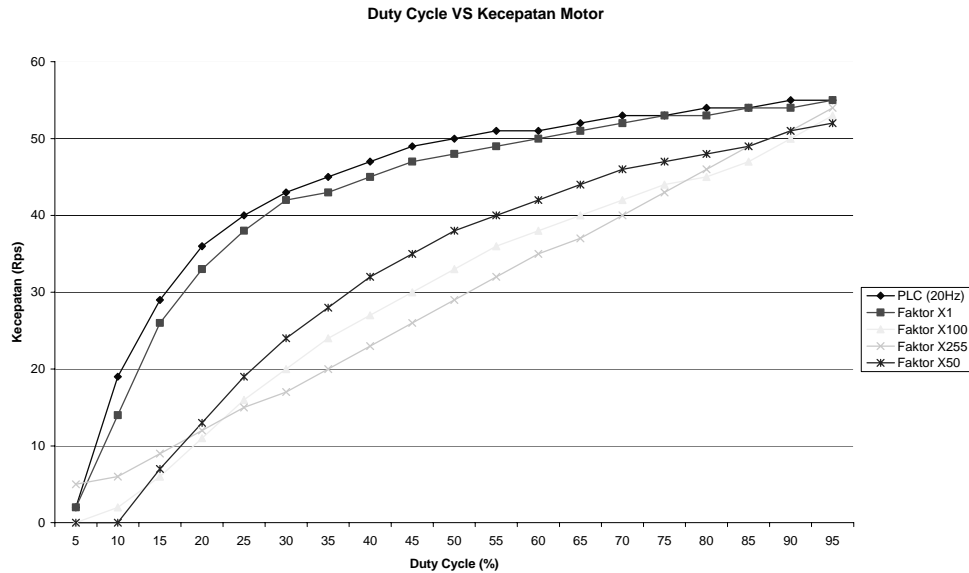


Fig. 11. Response of the motor for several multiplier factor

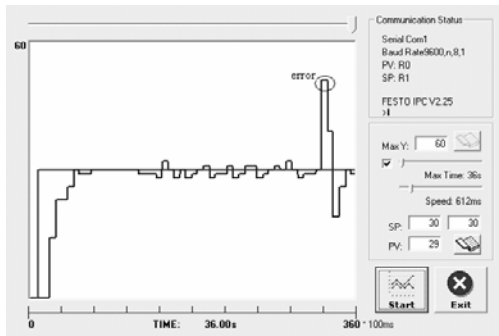


Fig. 12. Response of the motor with Fuzzy Logic Controller

Several experiments shows strange result, it looks like the controller loose control for some time (see figure 12). This is due to the microcontroller receives several interrupts but it can only serve with the highest priority. The highest priority is given to the measurement of the width of the pulse. Using higher crystal oscillator can help to fix this problem.

The error is also due to the missing tick when the system saves timer value to certain location. When the timer is stopped, it still receives clock ticks but the timer ignored it. So, when the timer value is saved, the system loses several ticks. This makes the measurement of the pulse width inaccurate.

#### REFERENCES

- [1] FESTO Didactic, *The CPU I/O Module FEC FC34-FST*, Germany: FESTO, 1999.

- [2] Christanto, Danny & Chris Pusporini, *Panduan Dasar Mikrokontroler Keluarga MCS51*, Surabaya: Innovative Electronics, 2004
- [3] Setiawan, Enggal, *Pembuatan Modul Fuzzy Controller Untuk PLC Festo*, Tugas Akhir S-1, Surabaya: Universitas Kristen Petra, 2003