

Introduction

The use of frequency or pulse output is well established as a means of communicating a flow rate or other measurement from a field instrument to a control system (Middelhoek 1988, Kiriniari 2000). Either the frequency or the duty cycle of a rectangular wave can carry information about the measurand variable (de Graaf 1997, Gummenjuk 1997). The technique usually offers better noise immunity, accuracy and resolution than the 4-20mA signalling standard that remains widespread within industry. Techniques have been developed for simultaneous transmission of multiple channels of data (Kirianaki 2000, Williams 1995).

Figure 1. illustrates one application of frequency output. The transmitter maps the current measurement value onto an equivalent frequency. A square wave of this frequency is generated and sent out over a dedicated pair of wires. At the control system, the signal is received and the frequency estimated, and the inverse of the mapping function is applied to recover the measurement value. More recently, smart transducers have been developed which use frequency output to communicate data (Kirianaki 2000).

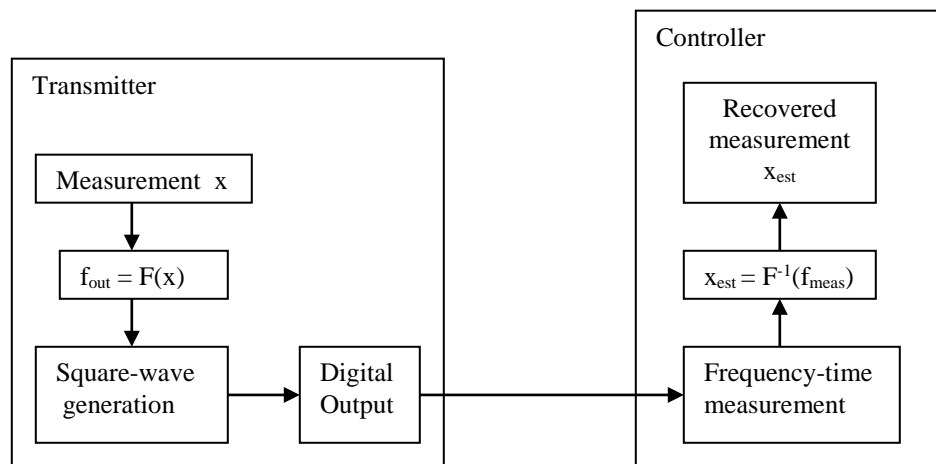


Figure 1. The transmitter converts the measurement to frequency; the controller recovers the measurement.

Recent years have seen rapid advances in the development of digital technology, with many implications for field instrumentation. Two of the most significant advances are:

- The employment of microprocessors of increasing computational power within instruments. This enables new functionality of increasing sophistication, such as linearisation of reading, compensation for secondary variables such as temperature, conversion to engineering units, self-calibration, diagnostics (Middlehoek 1988, Henry 2000), and sensor validation (Henry 1996, Tian 2001).
- The use of a digital communication protocol (called here a fieldbus), such as HART, Profibus or Foundation Fieldbus. By allowing two-way transmission of many variables (including measurement, configuration, and diagnostic data, device history etc), a fieldbus opens up the full potential of microprocessor-based instrumentation (Tian 2001, Control Engineering 2001).

The fieldbus provides numerous benefits compared with the conventional, one-parameter, one-directional 4-20mA and frequency techniques. For the purposes of measurement transmission perhaps the most important is the guarantee of full floating-point precision if required. For all practical purposes the

measurement uncertainty introduced by fieldbus communications is negligible, whereas with frequency and particularly 4-20mA the additional uncertainty can be appreciable. As instrument technology continues to improve, such uncertainties are likely to become substantially greater than the measurement precision associated with the instrument, and hence increasingly unacceptable. However, the fieldbuses themselves introduce new problems, including time delay, complexity, cost and technological uncertainty (Control Engineering 2001). A recent survey (Control Engineering 2001) found that 94% of respondents use 4-20mA as a communication protocol for transmitters, due to low cost and/or low measurement quality requirements. 35% used HART, 20% used Ethernet, and only 10% used a fieldbus.

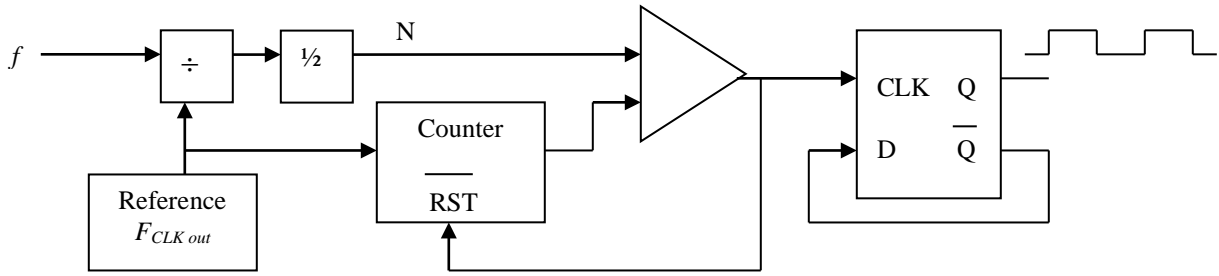
Although it is likely that fieldbus technology (of one flavour or another) will replace the one-directional methods of field communications over the next two decades, this will be a slow changeover and that 4-20mA and frequency will remain in widespread though declining use over the same time period. In the meantime frequency output from transducers is set to grow - Kiriniaki 2000 finds that 10% of smart transducers are now using this technique.

The purpose of this paper is to review the design issues associated with a digital implementation of frequency output, with an emphasis on the particular requirements for flow measurement, and to illustrate how, with the use of modern digital technology and hardware compilation tools, very high precision (i.e. 1 part in 10,000,000) frequency generation and collection can be achieved. Complete descriptions of frequency transmitter hardware are provided in the form of C-like software, which are readily adapted to the requirements of particular applications.

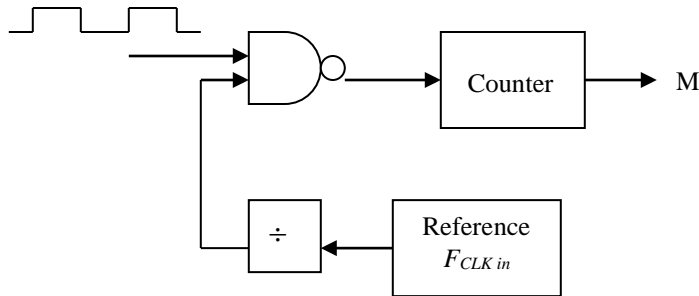
Digital Frequency Generation

Although the first implementations of frequency output were based on analogue circuitry, such as voltage-to-frequency converters (Burr-Brown 1994), integrated devices (Gumenjuk 1997, de Graaf 1997, Kiriniaki 2000), or fully digital implementations are becoming increasingly common (Hatfield 1999), due to the ever-reducing cost of digital hardware, ease of integration into ASICS, and the reduction or elimination of error sources such as component drift and temperature effects.

All digital components carry out actions synchronised to the rising or falling edges of a supplied clock frequency, F_{CLK} , a square wave signal typically supplied by a crystal oscillator and supporting circuit. In a contemporary design, F_{CLK} might vary from 100 kHz to 100MHz or more. Figure 2 shows a basic implementation of a digital frequency generation (a) and capture (b). In the frequency generator, a timing register (or timer) counts clock ticks. When the timer reaches the specified target value, the output line is inverted and the accumulator reset to zero. The continuous inversion of the output signal generates the frequency of the desired value. In the most common form of frequency capture, a register is used to count the number of input waveforms over a finite time interval, as measured by a timer. When the timer reaches the set interval t_{sample} , the number of waveforms is stored for transmission to the processor (by polling or interrupt), and the counter and timer are reset.



(a) Frequency generation



(b) Frequency measurement

Fig. 2 Simple frequency generation and capture

Both transmission and capture of frequency signals require a small number of simple digital operations (increment, comparison) and resources (e.g. memory), with action every clock cycle. It is thus inefficient to use a processor (however simple) for these tasks, and it is far more common to use dedicated digital hardware, usually interfaced to a processor, to relieve the burden of cycle-by-cycle operations.

The frequency range used for mapping the measurement value is constrained at both extremes. The lowest frequency that can be generated is limited by the size of the counter register. If this is of length k bits, then the maximum count is $2^k - 1$, and so the minimum frequency that can be generated is $0.5 * F_{CLKout} / (2^k - 1)$. Theoretically, the highest frequency that can be generated is simply $0.5 * F_{CLKout}$, corresponding to a target count value of 1. In most practical applications, however, the determining factor is the ability of the communication medium to transmit square wave pulses without significant distortion, and typical maximum frequency values are in the range 1-100 kHz.

Many methods have been developed over the last three decades for accurate or easy frequency measurement (Stein 1985, Burr-Brown 1994, Williams 1995, Kirianaki 1998, Kirianaki 2001), but there is less in the literature describing improved frequency generation. Commercial sensors and transmitters typically offer precision of 0.001%, at its best (Kirianaki 2000). Hatfield 1999 describes a nose sensor implemented using FPGAs, with a frequency output varying by less than $\pm 0.002\%$ when using a HP53132A frequency counter. Zhou et al. (Zhou 2000) developed a method of deleting clock ticks to achieve precisions of the order of 10^{-10} , but at expenses of a very complicated circuitry. This paper describes a simple frequency generation technique which, when implemented on low-cost hardware, provides a precision of $10^{-6}\%$.

Conventional method

In the most common approach to frequency generation the number of clock ticks *target_count* (or *N* in Fig. 2a) is computed to complete half a period of the desired frequency *f*. The value of *target_count* is given by

$$target_count = \text{round}\left(\frac{F_{CLKout}}{2f}\right), \quad (1)$$

where $\text{round}(x)$ is the nearest integer to x . A counter is incremented each clock tick and compared with *target_count*. When the target is reached, the pulse output is inverted and the counter is reset. Usually the method is implemented on a piece of dedicated hardware, as in Fig. 2a, but the following pseudo-code shows the procedure:

```
while TRUE {
    count=0;
    while (count < target_count) count++;
    invert_pulse_output();
}
```

The accuracy of this algorithm is primarily a function of how accurately the true clock frequency is known – the issue of crystal frequency calibration and compensation for temperature variation is well understood and not discussed further. Of greater interest is the precision of the algorithm i.e. assuming the true clock frequency is known, how precisely can an arbitrary frequency be generated?

The conventional method has a single parameter, *target_count*, which is an integer. The only frequency values that can be generated are thus

$$exact_f_{out}(target_count) = \left(\frac{F_{CLKout}}{2 \cdot target_count}\right), \quad (2)$$

For low output frequencies, *target_count* is high, and so the precision error is small. For higher output frequencies, the precision error grows rapidly.. For example, with a 40 MHz clock, to generate a 1 Hz signal, the number of clock ticks to complete half a cycle, *target_count*, is 20,000,000, so the frequency precision is 1 part in 20 million; for a 10 kHz signal, however, *target_count* is 2,000 and the resolution is only 1 part in 2 thousand. Adjacent generatable exact frequencies are 9,995 Hz and 10,005 Hz, and so precision error may be as high as 2.5Hz.

Dithering Technique

For a given clock frequency, each square wave pulse takes place over an integer number of clock ticks, and therefore can take one of only a few discrete frequencies. To represent an intermediate frequency, dithering can be used to provide a sequence of pulses which, averaged over some reasonably short timescale, provides the desired average frequency to high precision.

Two parameters are used: a variable step to increase the counter along with the conventional target value. Modulo arithmetic is used to restart the counter instead of a simple reset. The counter accumulator *accum*, initially set at zero, is increased on every clock tick by a quantity *step*. When the accumulator reaches or exceeded the value of *target_count*, the output is inverted and modulo arithmetic operation (MOD) of *accum* with *target_count* is performed, such that a variable remainder is left in the accumulator of the counter, as is shown in the pseudo-code that follows:

```
accum0 = 0;
while TRUE {
    while (accumt < TARGET_COUNT) accumt = accumt-1 + step;
    invert_pulse_output();
    accumt = accumt MOD TARGET_COUNT;
}
```

It can be seen that the value of $accum$ for the t th clock tick is:

$$accum_t = (t \cdot step) \text{ MOD } target_count \quad (3)$$

Hence, after $target_count$ clock cycles,

$$accum_t = (target_count \cdot step) \text{ MOD } target_count = 0 \quad (4)$$

Eq. 1.3 shows that the accumulator is reset to zero every $target_count$ clock cycles, and the number of output cycles in this time window is equal to $step$. Thus the output sequence repeats every $target_count$ clock cycles, generating exactly $step$ cycles. The two integer values, $step$ and $target_count$, are selected such that their ratio is as close as possible to $(2f / F_{CLKout})$:

$$\frac{step}{target_count} \approx \frac{2f}{F_{CLKout}} \quad (5)$$

As outlined below, $step$ and $target_count$ are selected such that $target_count$ is as large as possible for the given hardware. This provides a simple algorithm for their selection, while providing simple expressions for the maximum frequency precision error, and ensuring this error is small. The frequency precision error is given by:

$$\frac{step}{target_count + 1} < \frac{2f}{F_{CLKout}} < \frac{step}{target_count} \quad (6)$$

for a positive error, and:

$$\frac{step}{target_count} < \frac{2f}{F_{CLKout}} < \frac{step}{target_count + 1} \quad (7)$$

for a negative error. It is clear that the maximum error in f is inversely proportional to $target_count$. Therefore, in order to keep this error small, $target_count$ should be as large as possible.

For example, suppose that the clock frequency is 40 MHz and the desired f is approximately 10 kHz. If $target_count$ can be as high as 40,000,000, a $step$ value of 20,000 generates an exact frequency of 10 kHz at the output. Increasing $target_count$ to 40,000,001 generates a sequence of pulses with an average frequency over 40,000,001 clock cycles (i.e. 1 second) of 9,999.999,75 Hz. The available precision in f is thus 1 part in 40 million, i.e. 0.000,25 Hz.

The algorithm can be implemented in silicon more efficiently by assuming $target_count$ will be as close as possible to the largest power of two representable in the accumulation register, MAX_VAL . Their difference is denoted $offset$. Comparing the accumulated value with MAX_VAL is equivalent to testing its top bit; applying the modulo operator for MAX_VAL is equivalent to resetting the top bit of the accumulator. The modified algorithm is:

```
accum0 = 0;
while TRUE {
    while (accum_top_bit_not_set()) accumt = accumt-1 + step;
    invert_pulse_output();
    reset_accum_top_bit();
    accumt = accumt + offset;
}
```

Further simplifications are possible if the top bit of the accumulator can be used as the pulse output bit itself. In any real-time implementation, the value of *offset* may need augmenting by one or more times *step* in order to compensate for the clock cycles used to carry out the instructions for when the top bit is set.

The value of *step* is calculated using *MAX_VAL* as a rough approximation of *target_count* in Eq. 1.4.

$$\begin{aligned}
\frac{step}{MAX_VAL} &< \frac{2f}{F_{CLKout}} \\
\Rightarrow step &< \frac{MAX_VAL \cdot 2f}{F_{CLKout}} \\
\rightarrow step &= \text{floor}\left(\frac{MAX_VAL \cdot 2f}{F_{CLKout}}\right)
\end{aligned} \tag{8}$$

where $\text{floor}(x)$ denotes the integer part of x . Once *step* is calculated, the best value of *target_count* to approximate the output frequency f_{out} to the desired frequency f is:

$$\begin{aligned}
\frac{step}{target_count} &\approx \frac{2f}{F_{CLKout}} \\
\Rightarrow target_count &= \text{round}\left(\frac{step \cdot F_{CLKout}}{2f}\right)
\end{aligned} \tag{9}$$

Therefore, *offset* is:

$$\begin{aligned}
offset &= MAX_VAL - target_count \\
&= MAX_VAL - \text{round}\left(\frac{step \cdot F_{CLKout}}{2f}\right) \\
&= \text{round}\left(MAX_VAL - \frac{step \cdot F_{CLKout}}{2f}\right)
\end{aligned} \tag{10}$$

MAX_VAL limits the accuracy as well as the lower bound of f . Its value must be such that the quantity $(MAX_VAL \cdot 2f)$ is greater or equal than F_{CLKout} for the lowest frequency f , so *step*, as calculated in Eq. 1.7, is greater than zero for all frequencies in the range. The maximum frequency is obviously limited by F_{CLKout} . *MAX_VAL* also dictates the maximum sequence length and hence the minimum time period over which the theoretical frequency precision can be achieved, this being MAX_VAL / F_{CLKout} seconds.

Experimental Results

The modulo approach was implemented in Handel-C on a FPGA (Xilinx Spartan IIe) and tested for several frequencies in the range 2 Hz – 20 KHz. The output frequency was measured using a digital frequency meter (Agilent Universal Counter 53131A) with 10-digit precision, gate time of 2s, and 10 measurements for the statistics. Simulation and experimental results for 10 kHz, are presented in this section. Small variations in the desired frequency, of the order of 1 part in 10 million were introduced to test the precision of the method. Table I shows the values of the optimal parameters, *step*, *target_count* and *offset* for the desired frequency, using a 40 MHz clock and with *MAX_VAL* chosen to be 2^{24} . The theoretical value for the frequency output are given, computed as:

$$\begin{aligned} \frac{2f_{out}}{F_{CLKout}} &= \frac{step}{target_count} \\ \Rightarrow f_{out} &= \frac{step \times F_{CLKout}}{2 \times target_count} \end{aligned} \quad (11)$$

along with the value generated during simulation as the average over a window with *step* number of output cycles. Table I also shows the precision error due to the selection of integer values for *step* and *offset*. It can be seen that this error is always less than half of the quantization step, which at 10 kHz and for the chosen value of *MAX_VAL* is ± 0.0006 Hz.

For the experimental tests, the instrument error (typically $< \pm 0.0720$ Hz) and the FPGA crystal frequency error, mainly caused by slow temperature drift, were compensated for by adjusting the assumed value of F_{CLKout} such that f_{out} is as close as possible to f at 10 kHz, which in theory has a zero quantization error. The results can be found in Table II, including mean and standard deviation of the real frequency output f_{out} , as well as the absolute and the relative error with respect to f , computed with 2σ confidence.

Required frequency f (Hz)	Optimal parameters			Theoretical f_{out} (Hz)	Simulated f_{out} (Hz)	Quantization error (μ Hz)
	<i>step</i>	<i>target_count</i>	<i>offset</i>			
9,999.997	8388	16776005	1211	9,999.997,020	9,999.997,020	-20
9,999.998	8388	16776003	1213	9,999.998,212	9,999.998,212	-212
9,999.999	8388	16776001	1214	9,999.998,808	9,999.998,808	192
10,000.000	8388	16775999	1216	10,000.000,000	10,000.000,000	0
10,000.001	8388	16775997	1218	10,000.001,192	10,000.001,192	-192
10,000.002	8388	16775996	1219	10,000.001,788	10,000.001,789	212

Table I. Optimal parameters for desired frequency f , and theoretical and simulation values of frequency output f_{out}

Required frequency f (Hz)	Experimental results			
	Average f_{out} (Hz)	Standard deviation (μ Hz)	Frequency error (μ Hz)	Frequency error (%)
9,999.997	9,999.996,822,04	49.16	177.96 ± 98.32	$(1.8 \pm 1.0) \times 10^{-6}$
9,999.998	9,999.998,076,21	38.25	-76.21 ± 76.50	$(7.6 \pm 0.8) \times 10^{-7}$
9,999.999	9,999.999,379,71	25.23	-379.71 ± 50.46	$(-3.8 \pm 0.5) \times 10^{-6}$
10,000.000	10,000.000,203,8	21.21	-203.8 ± 42.4	$(-2.0 \pm 0.4) \times 10^{-6}$
10,000.001	10,000.001,297,0	27.59	-297.0 ± 55.2	$(-3.0 \pm 0.6) \times 10^{-6}$
10,000.002	10,000.002,430,7	34.65	-430.7 ± 69.3	$(-4.3 \pm 0.7) \times 10^{-6}$

Table II. Experimental results

For the given value of *MAX_VAL*, the theoretical precision is $6 \times 10^{-6} \% ^1$, which is achieved in the experimental data.

Conclusions

Measurements taken under laboratory normal conditions show that the dithering technique for digital frequency generation provides high precision, of the order of 10^{-7} can be achieved. This precision, constant for all the frequency range, depends in theory only on *MAX_VAL*; but in practice the true accuracy is limited by the stability of the frequency reference. Temperature fluctuations are the main contributors to crystal clock instabilities.

¹ Note: $\Delta f/f \approx 1 / MAX_VAL = 0.000,000,06 = 6 \times 10^{-8}$, or $6 \times 10^{-6} \%$

References

- Middelhoek, S., French, P.J., Huijsing, J.H., Lian, W.J. (1988), "Sensors with digital or frequency output", *Sensors and Actuators*, 15, pp.119-133.
- Burr-Brown (1994), "Voltage-to-Frequency converters offer useful options in A/D conversion, specialized counting techniques achieve improved speed and resolution", *Application Bulletin*, Burr-Brown, Tucson.
- Control Engineering (2001), "Product Focus: Process Variable Transmitters, PVTs use diagnostics, go digital, deliver multiple readings", *Control Engineering*, April 2001, pp. 69-74.
- de Graaf, G., Riedijk, F.R., Wolffenbuttel, R.F. (1997), "Colour-sensor system with a frequency output and an ISS or I²C bus interface", *Sensors and Actuators A (Physical)*, vol. A61, pp. 441-5.
- Gumenjuk, S., Podlepetsky, B. (1997), "Advanced integrated magnetic-field sensors with frequency output", *Sensors and Actuators A (Physical)*, vol. A62, no. 1-3, pp. 529-33.
- Hatfield, J.V., Daniels, A.R., Snowden, D., Persaud, K.C., Payne, P.A. (1999), "Development of a Hand Held Electronic Nose (H²EN)", in Bartek M. (Ed.), *Euroensors XIII. the 13th European conference on solid-state transducers proceedings*, pp. 215-218.
- Kirianaki, N.V., Yurish S.Y., Shpak N.O. (2001), "Methods of dependent count for frequency measurements", *Measurement*, vol. 29, no. 1, pp. 31-50.
- Kirianaki, N.V., Yurish S.Y., Shpak N.O. (2000), "Smart sensors with frequency output: state-of-the-art and future development", in *Programmable Devices and Systems (PDS 2000) Proceedings, IFAC Workshop*, Elsevier Science, Kidlington, UK pp. 37-42.
- Kirianaki, N.V., Yurish S.Y., Shpak N.O. (1998), "New processing methods for microcontroller compatible sensors with frequency output", in *Euroensors XII Proceedings of the 12th European Conference on Solid-State Transducers and the 9th UK Conference on Sensors and their Applications*, IOP Publishing, Bristol, UK, vol. 2, pp. 883-6.
- Stein, S.R. (1985), "Frequency and time, their measurement and characterization", in Gerber, E.A. and Ballato, A. (Eds.), *Precision frequency control*, Academic Press, New York, pp. 191-416.
- Tian, G.Y. (2001), "Design and implementation of distributed measurement systems using fieldbus-based intelligent sensors", *IEEE-Transactions-on-Instrumentation-and-Measurement*, vol. 50, no. 5, pp. 1197-202.
- Williams III, A., Fraenkel, N.R. (1995), "Simultaneous measurement of pressure, temperature, and conductivity with counters" in *Challenges of Our Changing Global Environment Conference Proceedings, OCEANS '95 MTS/IEEE, IEEE*, New York, vol. 1, pp. 626-30.
- Zhou, W., Li, Z., Zhou, H. (2000), "A practical method to process time and frequency signal", *IEEE Transactions on Ultrasonics Ferroelectrics and Frequency Control*, vol. 47, no. 2, pp. 480-3.