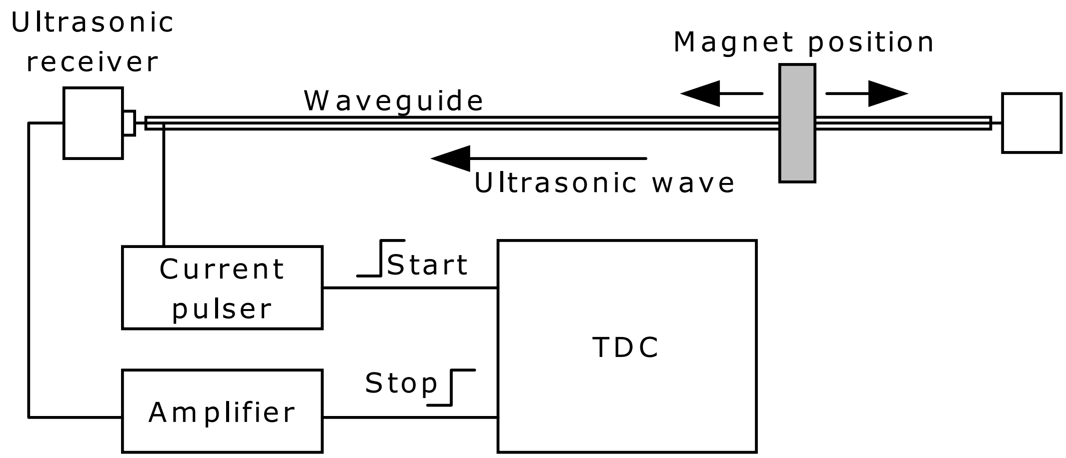
Position Sensing with Magnetostrictive Devices Improving Precision using TDCs

Non-contact, linear position sensing is necessary for many industrial applications. Magnetostrictive transducers in addition offer a very high precision of less than 0.001 inch. They are well established in applications like plastic injection molding machines, hydraulic and pneumatic cylinders or wood-working machinery. A basic challenge for the electronics is high-precision time measurement and can be easily solved usind TDCs (Time-to-Digital Converter)



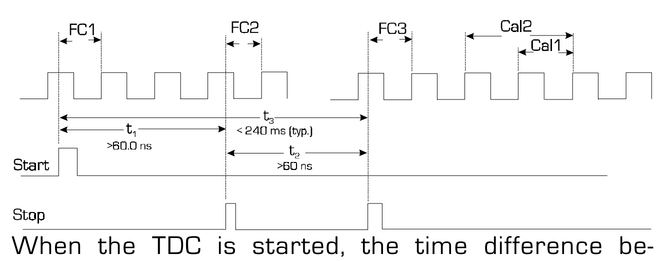
The Measuring Principle

The measuring element consists of a magnetostrictive waveguide. Magnetostrictive materials are elastically deformed when a magnetic field is present. This effect is used in the following manner:
The magnetostrictive waveguide is built as tube with a copper rod inside. The start of measurement is initiated by a short current pulse. This produces a circular magnetic field around the waveguide. The position of the movable part is marked by a magnet whose magnetic field is perpendicular to the circular field of the current pulse. The interaction between the two magnetic fields produces a strain pulse, which travels at sonic speed along the waveguide. A sensor placed at the end of the waveguide converts the sonic pulse into an electrical signal. The travel time is directly proportional to the position of the magnet. The sonic speed in the waveguide is approximately 2830m/s, which corresponds to approx. 9µs/inch or 0.35µs/mm. To achieve a resolution of 0.0002 inches (5µm), the precision in time measurement must be 1.8ns! This corresponds to a reference clock in the GHz range.

Time Measurement with TDCs

TDC-GP1 is a good solution for the time measurement task in magnetostrictive applications. This single-chip time-to-digital converter has 2 channels with a single shot resolution of 250ps and therefore easily fullfills the needs of high precision positioning. Furthermore, the 30bit dynamic range of TDC-GP1 allows measurements up to 200ms with the same resolution. With it's 4-fold multihit capability up to 4 magnets can be handled at once. TDC-GP1 is available in a small TQFP44 package allowing compact board design.

Several different measurement modes are available with TDC-GP1. For magnetostrictive applications the one called ‚measurement range 2' is the right choice. In this mode, the TDC uses a predivider to extend the measurement range from 7.6µs to 200ms.



When the TDC is started, the time difference between start and the next rising edge of the calibration clock is measured by the measuring unit with 250ps resolution (FC1 - finecount1). Afterwards a 16 bit counter is activated that counts the periods of the predivider ((cc – coarse count). The time between stop pulse and next rising edge of the reference counter is measured again with 250ps resolution (FC2 – finecout2). In the following, one and two periods of the calibration clock are measured for calibration (cal1, cal2).

The final time is calculated as:

$$time = \text{Tref} * 2^{\text{SEL\_CLK\_TDC}} * \left(cc + \frac{FC1 - FC2}{Cal2 - Cal1} \right)$$

cc=value of the precounter
T_{ref} = Period of reference clock (500kHz – 35MHz)
SEL_CLK_TDC = programmable value (0 … 6)

The data format is a 32 bit floating point number, that has to be multiplied by $T_{ref} * 2^{SEL\_CLK\_TDC}$. For data transfer the TDC-GP1 has a standard 8-bit interface.

Resolution

The resolution (LSB) of the TDC is 250 ps over the total measurement range. This corresponds to a displacment of

Resolution (LSB) = 0.000028 in (0.7 µm)

Measurement Range

The maximum of the measurement range is defined as:

$$T_{max} = T_{ref} * 2^{SEL\_CLK\_TDC} * 2^{16}$$

with ($2 * T_{ref} * 2^{SEL\_CLK\_TDC} < 7.6$ µs)

> Example:
> Working with a 8 MHz reference clock, Tref is 125ns. Setting SEL_CLK_TDC to 0, the maximum time is 8.192 ms.
>
> For magnetostrive positioning this means that the maximum rod length is > 905 in (23 m)

The minimum of the measurement range is defined as:

$$T_{min} = 1.5 * (T_{ref} * 2^{SEL\_CLK\_TDC}) + 25 \text{ ns.}$$

> Example:
> Working with a 8 MHz reference clock, Tref is 125ns. Setting SEL_CLK_TDC to 0, the minimum time is 212,5 ns.
>
> For magnetostrive positioning this means the minimum distance is 0.024 in (601 µm).

Multihit capability

In measurement range 2 only one stop channel is available. It's the stop 1 input that is enabled. This channel has 4-fold multihit capabilities in normal resolution mode. Always the time between stop and start is measured.

The double pulse resolution is given by:

$$T_{dp} = 1.5 * (T_{ref} * 2^{SEL\_CLK\_TDC}) + 25 \text{ ns.}$$

> Example:
> Working with a 2 MHz reference clock (500ns) and setting SEL_CLK_TDC to 0, the double pulse resolution is 850 ns.
>
> For magnetostrive positioning this means that the minimum distance between two magnets is 0.094 in (2.4 mm)

Programming TDC-GP1

In the following we'll show how to programm the TDC-GP1 in two different applications.

Measurement example 1:

Lets take a simple magnetostrictive system with only one magnet, that cannot be removed. The reference clock shall be 2MHz.

The program is split into two main blocks: the configuration routine, where the control registers are set, and the measurement routine.

Configuration:

```
output_byte(Reg7, 0x0)          Disbale inputs while writing to control registers
output_byte(Reg0, 0x78);        Set Measurement range 2, use autocalibration
output_byte(Reg2, 0x21);        Calculate Stop1 – Start
output_byte(Reg4, 0x00);        Set SEL_CLK_DIV to 0
output_byte(Reg6, 0x01);        Set Interrupt to ‚ALU ready‘, medium ALU-speed
output_byte(Reg7, 0x02);        Enable 2 hits on channel 1 (Start and Stop1)
```

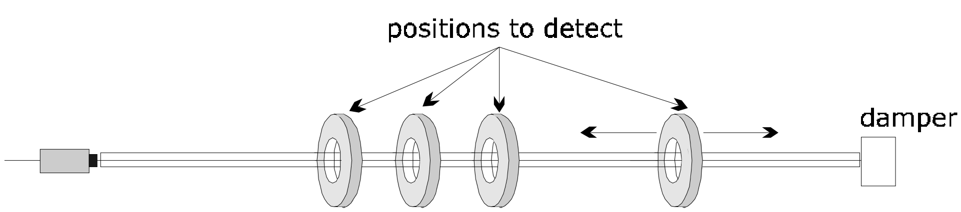
Measurement:

```
while(!quit)
{
        output_byte(Reg11,0x07);                Init TDC and ALU, Now the TDC is ready to measure
        valid = 0;
        while(valid==0)
        {
                valid=input_byte(INTFLAG);      Check Interrupt Flag, if ‘1’, data are available in
                                                output register
        }
        nk0=input_byte(Reg0);                   Lower byte  xxxx.xxXX
        nk1=input_byte(Reg0);                   Higher byte xxxx.XXxx
        vk0=input_byte(Reg0);                   Lower byte  xxXX.xxxx
        nk0=input_byte(Reg0);                   Higher byte XXxx.xxxx

        vk1=vk1*256;                            shift left
        vk1=vk1|vk0;                                    XXXX.xxxx
        nk1=nk1*256;                            shift left
        nk1=nk1|nk0;                                    xxxx.XXXX

        result = 500.0*((float)(vk1)+(float)(nk1)/65536.0);    Result in ‘ns’. 2 MHz reference clock -> 500 ns
        position = result + 0.005570866;                       Position in inch
        printf("position = %6.3f in\n",result);                Print position in inch
}
```

Measurement example 2:
Lets take a complex magnetostrictive system with four magnets. It may be possible to remove any of the magnets. The reference clock again shall be 2MHz.



Configuration:

```
output_byte(Reg7, 0x0);         Disbale inputs while writing to control registers
output_byte(Reg0, 0x78);        Set Measurement range 2, use autocalibration
output_byte(Reg2, 0x21);        Calculate Stop1 – Start, Position of the first magnet
output_byte(Reg4, 0x00);        Set SEL_CLK_DIV to 0
output_byte(Reg6, 0x81);        Set Interrupt to ‚Overflow‘, medium ALU-speed
output_byte(Reg7, 0x02);        Enable 4 hits on channel 1 (Start and Stop1-3) and
                                1 hit on channel 2 (Stop4, this hit is measured on
                                channel 1, but for internal storage, the registers of
                                channel 2 are used)
```

Measurement:

```
while(!quit)
{
        output_byte(Reg11,0x07);                              Init TDC and ALU
        valid = 0;
        while(valid==0)
        {
                status = input_byte(0x8);                     Read Statusregister 1
                if (status | 0x48)  > 0  then                 Check, whether the TDC could measure 4 hits
                {
                        valid =1;                             or if magnets are missing (overflow) }
                        n_hits = status | 0x4 + (staus | 0x8)/8;   number of hits (= number of magnets)
                }
        }
        adr_pointer = 2;
        hitx =  0x21;                                         number of hit, 0x21 = stop1, 0x31 = stop2 ...

        For (i=1,n_hits,i++)                                  Calculate all hit x – start time differences
        {
                valid=0
                while(valid==0)
                {
                        status = input_byte(0x9);             Read statusregister 2
                        status = stauts | 0x03;               Get pointer position in output register
                        if (status==adr_pointer) then         data are available in output register
                                valid=1;
                }
                adr_pointer = adr_pointer + 2;
                hitx = hitx + 0x10;                           Register 2 value for calculation of next hit
                output_byte(Reg2, 0x21);                      Start calculation of next hit (magnet)
        }

        For (i=1,n_hits,i++)                                  Data output for all hits
        {
                nk0=input_byte(Reg0);                         Lower byte  xxxx.xxXX
                nk1=input_byte(Reg0);                         Higher byte xxxx.XXxx
                vk0=input_byte(Reg0);                         Lower byte  xxXX.xxxx
                nk0=input_byte(Reg0);                         Higher byte XXxx.xxxx

                vk1=vk1*256;                                  shift left
                vk1=vk1|vk0;                                          XXXX.xxxx
                nk1=nk1*256;                                  shift left
                nk1=nk1|nk0;                                          xxxx.XXXX

        result = 500.0*((float)(vk1)+(float)(nk1)/65536.0);   Result in 'ns'. 2 MHz reference clock -> 500 ns
                position = result * 0.005570866;              Position in inch
                printf("position %d = %6.3f in\n",i,result);  Print position in inch
        }
}
```

1. Edition June 2001