Measuring The Output Signal of The Universal Transducer Interface Integrated Circuit with Microcontrollers

Stoyan N. Nihtianov and Ivan I. Sadinski

At the output (pin 12) of the Universal Transducer Interface (UTI) chip there is a continuous sequence of rectangular pulses (see Fig.1). The time intervals T_{xi} (for different modes these time intervals could be 1, 2 or 3) and the time interval T_{ref},

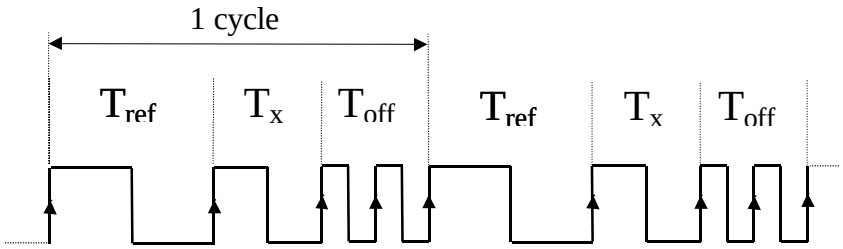

Fig.1 The output signal of the UTI.

between two adjacent rising edges, carry information about the values of the measurand(s) M_x (in Fig.1 only one measurand time interval T_x is shown) and the reference element M_{ref}, connected to the "C", "D", "E" and "F" pins of the UTI. The time interval T_{off} is presented by the sum of two periods. This time interval

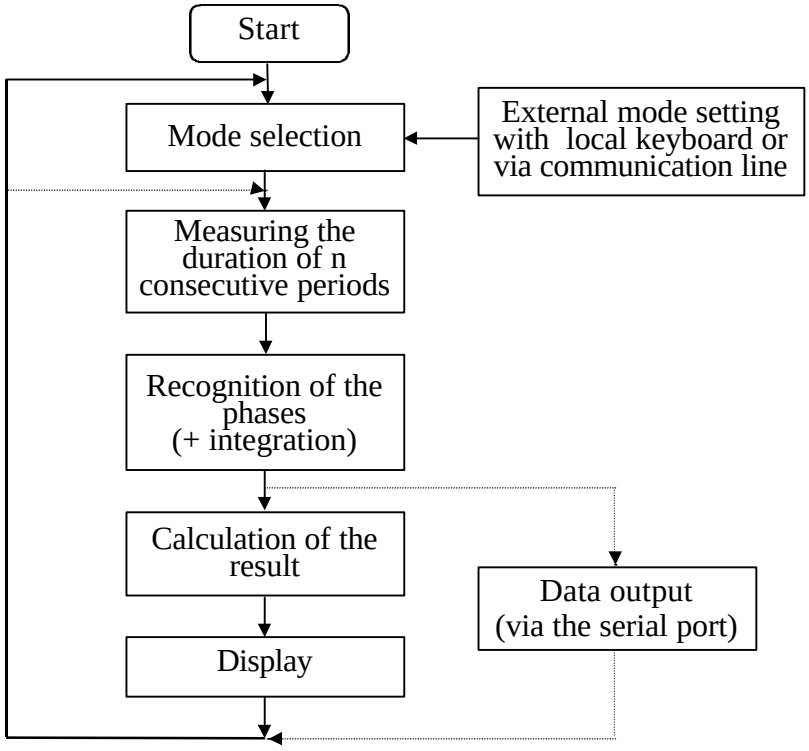

Fig.2.2 Flow chart of the UTI signal processing with microcontroller.

corresponds to the measurement of an element connected to pin “B” of the UTI and is usually applied for an offset measurement. The order of the phases is presented in Fig.1. Figure 2 shows the most important steps that have to be taken in the microcontroller in order to apply the three-signal method to obtain a measurement result with automatic autocalibration.

Mode selection. The mode of the UTI is selected by setting an appropriate level (0 or 1) at pins 4, 5, 6 and 7. This procedure also defines the number **n** (n = 3, 4 or 5) of the phases in one measurement cycle. One should keep in mind the fact that the number **M** of the periods necessary to obtain one measurement result is always with one bigger than the number of the phases: M = n + 1. This is because two periods are assigned to T_{off} . It is possible to change the mode after each measurement, in order to measure, for example, two or more different types of sensors in consecutive way. In this case some analogue input switches will have to be controlled, as well. One may choose, instead of expensive analogue switches, to use a separate UTI for each sensor. The power-down mode (pin 11) of the UTI chip allows a number of UTI outputs to be connected to one input of the microcontroller. The mode selection can be done automatically in the microcontroller programme. For other applications it will be more useful to change the mode with external mode setting (see Fig.2).

Measuring the duration of a number of consecutive periods. The minimum number of the measured consecutive periods, needed to obtain one measurement result, is M = n + 1. It is also possible to measure **k** successive cycles (the total number of the periods in this case will be L=k.(n+1)=k.M) and then to carry out some integration, before obtaining one measurement result. This can be easily accomplished by a variety of 8-bit low-cost microcontrollers, such as: Intel 8xC51, Texas Instruments TMS370Cxxx, Motorola 68HC711xx, Microchip PIC16Cxx/17Cxx, Hitachi H8/325, National Semiconductor COP884CG, SGS-Thomson ST90E27, etc..

The measurement of a number of sequential periods can be carried out in different ways. What is needed is a counter that continuously is counting the clock pulses. The counter has to be read each time, when a rising edge is detected of the UTI pulses. A simple way to measure periods is to use the available “capture” mode of the programmable timer/counter(s) of the above mentioned microcontrollers. The measurement procedure starts at an arbitrary rising edge of the incoming pulses. The positive transition of the input signal triggers a “capture” of the 16-bit current value of the timer/counter, which is automatically loaded into the 16-bit capture register and an interrupt, if enabled, is flagged. The difference between the previous and the current “captured” value of the counter is corresponding to the last measured period. This difference and the current “captured” value are stored in the memory and another capture event is expected. In this mode care should be taken for the overflows of the counter, which are important for the subtraction procedure. A peculiarity of some microcontrollers (for example Intel 8xC51) is that the capture events and the overflow of the counter share one and the same interrupt vector. Although seldom, it may happen so that the capture event and the overflow event occur within a very short period of time and both, the capture flag and the overflow flag, are raised in one interrupt procedure. In this case it is important to know which event has happened first, so that the subtraction is carried out in a proper way. There are different possibilities to solve this problem. For example, if the overflow is first - the stored value in the capture register and the current value of the counter (when read immediately after the interrupt procedure starts) will differ only in the Least Significant Bits (CAPL, TMRL) and not in the Most Significant Bits (CAPH, TMRH). If the

capture event is first, there will be also a difference in the Most Significant Bits of the two values (CAPH, TMRH).

Recognition of the phases and integration. To recognize the different phases out of the incoming rectangular pulses (see Fig.1), the two short periods, corresponding to the measurement of the input signal at terminal "B", are used. Usually this terminal is only applied for an offset measurement. In this way, even when the values of the other sensor signals are very small and almost equal the offset signal, the time periods T_{xi} , corresponding to this phases, will be approximately twice as large as any of the two T_{off} periods.

Different techniques can be applied to carry out the recognition procedure. If, first, **L** periods are measured and stored in consecutive way in the RAM memory and only after that the integration (summation) is carried out, then, within the first **M** stored periods (M is the number of periods in one cycle), the two shortest are looked for. Their addresses are enough to define the addresses of all the rest of the measured periods and then the summation of the corresponding phases from different cycles is easily accomplished (see the enclosed Example for one measurement procedure).

Another possibility is to carry out the summation during the measurement procedure. To do this, an additional counter has to be used to count the measured periods in every cycle so, that the corresponding periods from different cycles to be summed and stored in one and the same address. At the end we have (n+1) integrated results, corresponding to the **n** phases of one cycle. Before calculating the final result (by applying the auto-calibration three-signal method), the different phases have to be recognized by finding the two smallest results corresponding to T_{off}.

The calculation of the result, the display, and the data transfer, are routine procedures and are not going to be discussed here.

An example for one measurement procedure.

Microcontroller resources used:

PSW ⟦ov⟧ ⟦ca⟧ - processor status word (event flags),
PSW⟦ov⟧ - timer overflow,
PSW⟦ca⟧ - capture on positive edge.

CAPH
CAPL

- 16-bits capture register

TMRH
TMRL

- 16-bits timer

CAPH:=TMRH and CAPL:=TMRL - on positive edge of the input signal

BUF[0]			
BUF[*i*-1]	ov	hi	lo
BUF[*i*]			
BUF[*i*+1]			
BUF[L]			

- memory buffer,
BUF[*i*].ov - counts timer overflows between (*i*-1) and *i* capture events;

BUF[*i*].hi - time of *i* MSB;
BUF[*i*].lo - time of *i* LSB.

1. Measurement.

```
PSW [ov] := 0                          - set the timer running
PSW [ca] := 0                          - enable capture on positive edge
BUF [0]  := 0
        i := 0

WHILE i < L                            - L is the number of the measured periods

      x := PSW                         - snapshot of event flags
IF  x[ov] =1  and  x[ca] = 0  and  i≠0 - if overflow only
OR x[ov] =1 and  x[ca] =1  and  CAPH=0  and  i≠0  - or overflow before cupture
    BUF[i].ov = BUF[i].ov + 1          - count overflow

IF  x[ca] =1                           - upon capture
    (a) BUF[i].hi := CAPH              - store captured time
        BUF[i].lo := CAPL
    (b) BUF[i-1] := BUF[i] - BUF[i-1]  - capture period ⟨i-1, i⟩
    (c) i := i + 1                     - count positive transition

IF x[ov] =1 and  x[ca] =1  and  CAPH≠0  and  i≠L   - if capture before overflow
        BUF[i].ov := BUF[i].ov + 1     - count overflow

upon exit BUF contains L periods
```

2. Phase recognition.

```
     x := 0                            - x is the shortest period index
FOR  i = 1  to  M-1                    - M is the number of periods in 1 cycle
IF  BUF[x] > BUF[i]                    - try to find shorter period
THEN   x := i;                         - BUF[x] is the shortest period
IF BUF[x] < 0.8 BUF[(x+1)mod M]        - the second short period
    x := (x-1)mod M                    - have one back
```

3. Integration of (K cycles) x (M periods).

```
FOR  i = 1 to K-1                      - for all cycles
FOR  j = 0 to M-1                      - for each period
      BUF[j] := BUF[j] + BUF[M*i + j]
```

4. Output.

```
FOR i = 0 to M-1
      OUTPUT BUF[x]
      x := (x+1)mod M
```