1.2 GHz MULTIFUNCTION FREQUENCY METER

PART 4 (FINAL): THE PC LINK (CONTINUED) AND MEASUREMENT PRINCIPLES

Design by B.C. Zschocke

The PC may not transmit characters to the counter while this is busy executing the command string. Any character, in particular US, has the same effect as pressing the BREAK key on the instrument: it halts the execution of the command string, and takes the counter back to its start state (default). This may, of course, be used to break off a measurement on purpose.

By transmitting a **DC3** character, the PC prompts the counter to transmit the contents of all registers (Fig. 10i).

Control function **DC4** is used by the PC to read the current command stored in the counter (Fig. 10j). The return transmission starts with the first function contained in the command string. An ACK code indicates that the complete command has been transmitted. If the DC4 is followed immediately by ACK, the command memory is empty.

All functions contained in a command may also be executed directly, one by one (Fig. 10k). This is achieved by having the computer send the function to the counter (in connect mode). This is particularly useful when toggle-type settings such as buzzer on/off are to be changed.

A command string consists of a number of functions arranged as a sequence. On changing to command entry mode (STX), a pointer in the counter points to the first function in the command string (Fig. 10i). Any function sent to the counter is then added to the command string at the pointer position. Next, the pointer is increased by one. To check this loading process, the counter returns a copy of the stored character to the PC. If the command memory is full, the next function received is not stored, and a **GS** code is sent to the PC. Control function HT causes the function at the pointer position to be returned to the PC, and increases the pointer by one without storing the function. Control function **CAN** moves the pointer back one location, and transmits the character at the new location.

The counter returns a **NAK** code if it receives anything it can not interpret (i.e., any unknown control character or function) — see Fig. 10m.

The **RS** code allows the PC to reset the counter (Fig. 10n). This function is equivalent to switching the counter off and on again. After a reset, all register contents are undefined.

A **US** code, finally, causes the counter to revert to its start (default) state (break, Fig. 10o). At the same time, it leaves the connect or command entry mode.

Commands

A command consists of a number of individual indicators. The PC should build the command string in accordance with the structure of the menu overview shown in Fig. 8 (part 2). That is, from the top (reset) to the bottom (exit and start), with the functions preceding the parameters. As already mentioned, the relevant codes may be found in the boxes shown in Fig. 8. Table 3 lists all functions and associated codes.

An example is in order at this point to illustrate how a command string may be built. Let us assume that the following measurement is required.

Type:	frequency on channel A;
Gate time:	1 s;
Start on:	START key.

The string is shown analysed in Fig. 11. Also refer back to Fig. 8 to understand how the PC follows the menu structure. The two-position hexadecimal numbers are transmitted to the counter as one byte. The number of bytes per command is not fixed, since it is possible, as shown by the exam-

```
21 2F 57 3C 65 77 7F
                    └6th. parameter (setting: period analysis on)
                 └5th. parameter (setting: single)
              └4th. parameter (start on: START key)
           └3rd. parameter (gate time: 1 s)
        └2nd. parameter (gate: gate time)
     └1st. parameter (input: channel A)
 └function: frequency measurement
```

Fig. 11. Example of a command string sent to the frequency meter by the PC.

Table 4. Function descriptions

MAIN FUNCTIONS

Frequency:	frequency measurement
1/Frequency:	frequency measurement, reciprocal indication
Revolution counter:	frequency measurement, indication in rev./min

Frequency measurement requires the following parameter functions:
- Channel...(input)
- Gate...(measurement duration)
- Gate time...(gate time, including 'gate time measured')
- Start...(start of measurement)

Pulses: count pulses

Parameter functions required:
- Channel...(input)
- Gate... (measurement duration)
- Gate time...(gate time, only with 'Gate preset')
- Start...(start of measurement)

Time: time measurement

Parameter functions required:
- Gate on key pressed or start/stop key

Timer: timer device

Parameter functions required:
- User def. period (loads time to be set)
- Start...(start of measurement)
- Pulse...(pulse on output)

Pulse generator

Parameter functions required:
- Preset period duration or preset pulse/pause
- User def. period or
- User def. duration and user def. pause
- Start...(Start)
- End...(Stop)
- User def. pulse (with 'end on number of periods')

Zero counter

Zero counter/manual counter

Parameter functions required:
- Start value... (start value)
- Channel C L-H or channel C H-L (input)
- Active... or pulse on EQ0 or count pulse (output)
- Start on...(Start)

PARAMETER FUNCTIONS

Signal input

Channel A signal input:	channel A
Channel B signal input:	channel B
Channel C signal input:	channel C
Channel C H signal input:	channel C, active high-phase
Channel C L signal input:	channel C, active low-phase
Channel C H-L signal input:	channel C, active H-to-L edge
Channel C L-H signal input:	channel C, active L-to-H edge

Gate time (Gate)

Gate time 0.1 s	Load gate time register with 0.1 sec.
Gate time 1 s	Load gate time register with 1 sec.
Gate time 10 s	Load gate time register with 10 sec.
Gate time 1 min	Load gate time register with 1 min.
Gate time user def.	Gate time to equal contents of gate time register
Gate time measured	Gate time measured
Gate time rising	Gate time increases with each measurement

Preset times for pulse generator

Preset period duration	Period duration is preset (in period duration register)
Preset pulse/pause	Pulse duration and pulse pause are preset in corresponding registers
User def. period	Load period register
User def. duration	Load pulse duration register
User def. pause	Load pulse/pause register
User def. pulse	Load register with number of pulses

Start value

Start value nought	Preset start value register with 0 and load.
Start value user def.	Load start value register

Measurement duration (Gate)

Gate preset	Gate time defined by gate time register
Gate channel C high	Gate time defined by high pulse on channel C
Gate channel C low	Gate time defined by low pulse on channel C
Gate channel C H-L	Gate time defined by H-to-L transition on channel C
Gate channel C L-H	Gate time defined by L-to-H transition on channel C
Gate key pressed	Gate time as long as START key pressed
Gate start/stop key	Gate time starts on START key, and ends on STOP key

Start on (Start)

Start immediately	Start measurement/pulse generator immediately
Start on signal	Start measurement on detection of signal
Start on signal A	Start measurement/pulse generator on detection of signal on channel A
Start on signal B	Start measurement/pulse generator on detection of signal on channel B
Start on channel C H-L	H-to-L signal transition on channel C starts measurement/pulse generator
Start on channel C L-H	L-to-H signal transition on channel C starts measurement/pulse generator
Start on START key	Start key starts measurement/pulse generator

End on (Stop)

End on no. of periods	End after predetermined number of periods
End on channel C H-L	End on H-to-L transition on channel C
End on channel C L-H	End on L-to-H transition on channel C
End on STOP key	End when STOP key pressed

Output (Timer)

Pulse on start/end	Output pulse on start and end
Pulse on start	Output pulse on start
Pulse on end	Output pulse on end
Pulse f. start to end	Output active from start to end

Output (Manual/Zero counter)

Active when NE0	Output active as long as count $\neq 0$
Active when EQ0	Output active as long as count = 0
Pulse on EQ0	Output pulse when counter reaches state 0
Count pulse	One output pulse per count pulse

The following functions may be executed directly or as parameter functions:

Measurement order	Change from 'continuous' to 'single' and the other way around.
Continuous	Continuous series of measurements
Single	Single measurement
Buzzer	Toggle buzzer on/off
Beep on	Switch on buzzer
Beep off	Switch off buzzer
Intermediate value	Switch between 'interm. result displayed' and 'interm. result not displayed' (toggle).
With interm. result	Switch on intermediate result display function
Without interm. result	Switch off intermediate result display function
Period analysis	Change between 'Period analysis on' and 'Period analysis off' (toggle)
Period analysis on	as is
Period analysis off	as is
Pulse polarity	Change pulse polarity
Pulse polarity pos.	Pulse polarity is positive
Pulse polarity neg.	Pulse polarity is negative
Inactive level	Change pulse inactive level
Inactive level low	Pulse inactive level is low (0 V)
Inactive level high	Pulse inactive level is high (+5 V)
Main Break	Do BREAK (counter changes to basic mode/settings)
Reset	Do RESET
Run command	Execute command
Buzzer	Beep!

REGISTER DESIGNATIONS (all values unsigned)

Direct registers (R1,..., Rn)

R1	Frequency measurement: real measurement (gate-) time in µs (after measurement) Pulse counter: real measurement (gate-) time in µs (after measurement) Time measurement: measured time in µs Pulse generator: preset pulse duration in µs Timer: remaining time in µs
R2	Pulse generator: period duration (if preset) in µs Timer: preset time in µs Manual/Zero counter: start value
R3	Frequency measurement: number of pulses counted, (prescaler ignored). Measured frequency computed from: R3/R1 Pulse counter: number of pulses counted (prescaler ignored) Pulse generator: preset pulse pause duration in µs Manual/zero counter: counter state
R4	Reserved
R5	Pulse generator: number of pulses to be generated

Indirect registers (I1,...,In)

I1	Reserved
I2	Frequency measurement: result (as on LCD readout) without exponent
I3	Reserved
I4	Reserved
I5	Frequency measurement: preset time in µs

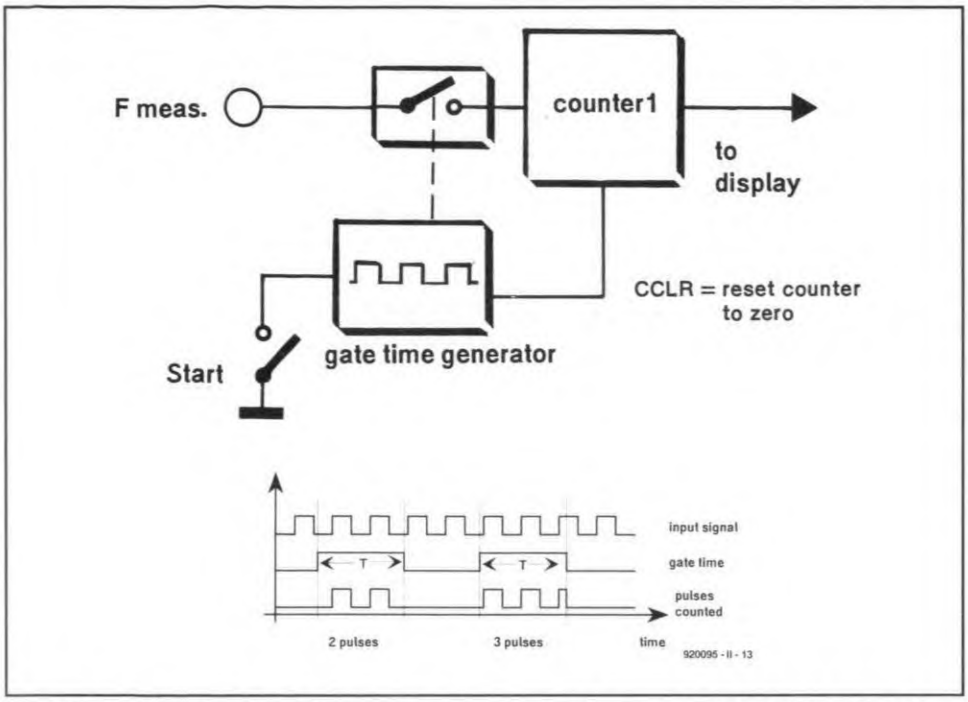


Fig. 12. The 'classic' digital frequency meter counts the periods of the input signal for a predefined time.

ple, that more than one parameter is required to complete the **settings**.

The counter executes the command from the right to the left, i.e., the **settings** before the **functions**.

You may not use parameters that are shown without a box code (Fig. 8), or that are marked with an asterisk in Table 3. Since the counter does not run a 'plausibility check' on received command strings, the user must make sure that these consist of meaningful parameters. This is not difficult to ensure by virtue of a useful trick that may be used during program development: simply use manual control to give the instrument the desired settings, and then read out the string using the DC4 command.

The frequency measurement principle

In an earlier instalment of this article it was stated that a separate instalment was to be devoted to the frequency measurement principle used by the instrument. The division of the complete article into instalments having taken a slightly different form than originally planned, we have decided to include this information in the present (final) instalment.

The usual way of measuring frequencies is to count the number of pulses that occur within a predefined gate time. Although this is not the most accurate method, it is by far the simplest. The number crunching power of a microcontroller or microprocessor, however, allows us to devise much more advanced measurement methods, of which practical applications may be found in Refs. 1 and 2. Although the same measurement principle is used for the frequency meter function of the 1.2-GHz multifunction frequency meter, this instrument makes even more use of combined software and hardware possibilities offered by the microcontroller. Also, there are now two counters instead of one counter and a programmable divider (of which the setting is determined beforehand by running a 'sample' measurement). The second counter replaces the programmable divider (in digital design, dividers and counters are often considered identical components). The nice thing about this new setup is that the sample measurement is no longer required, which results in a shorter measurement time. To understand how this works, it may be useful to recap the operation of the pulse counting principle used in 'classic' frequency meters.

The classic approach

To refresh your memory, Fig. 12 shows the architecture of the classic, pulse counting, frequency meter. A clock circuit supplies a gate signal that serves to connect the input signal to the counter for an accurately determined time, *T*. The number of pulses *N* counted during the gate time *T* thus gives the input signal frequency ($F=N/T$).

The accuracy of the measurement is determined by two factors: first, the accuracy of the gate time, and, secondly, the number of pulses counted. The latter factor is responsible for the relatively low accuracy at low frequencies. As illustrated by the timing diagram in Fig. 12, there may be an error of one in the number of pulses counted. As shown, it all depends on how the gate time, *T*, coincides with the periods of the input signal. The resulting absolute error, Δ_{abs}, is calculated from

$$\Delta_{abs} = 1\ \text{(pulse)} \;/\; T\ \text{(s)}\ \ \text{[Hz]}$$

Consequently, the measured frequency may have an error of 1 Hz at a gate time of 1 s, and 10 Hz at a gate time of 0.1 s. This error becomes smaller as the frequency increases, when the main cause of errors is increasingly on account of gate signal deviations. The table below shows the effect of the counting error at a gate time of 0.1 s:

f	Δ_{abs}	Δ_{rel}
1 MHz	10 Hz	0.001%
1 kHz	10 Hz	1%
10 Hz	10 Hz	100%

Frequencies lower than 10 Hz are not given simply because they can not be measured at a gate time of 0.1 s. Inevitably, lower frequencies require longer measurement times, which brings us to another disadvantage of the classic frequency measurement principle: measuring low frequencies accurately takes a lot of time.

Ratio-based measurements

A measurement principle that is eminently suited to microprocessor implementation is shown in Fig. 13. The basic principle is very simple. A certain time is reserved to measure the periods of the input signal and those of the reference frequency. Dividing the two gives the ratio of the input frequency and the reference frequency. Multiplying this ratio with the reference frequency then yields the frequency of the input signal.

If we say "a certain time", this has to be taken literally, since the gate time is really only an auxiliary signal in this setup. The input signal frequency is calculated exclusively on the basis of the counter states *N* :

$$f = f_{ref}\,(N_1 / N_2).$$

Bear in mind, however, that the measurement has to run for at least one period of the input signal.

The fact that the gate time is an independent parameter, opens up the possibility to use the computer for 'fine tuning' of the result, or, in other words, make the measurement a little more accurate. This is necessary anyway because both counters make an error of one pulse if the gate time were

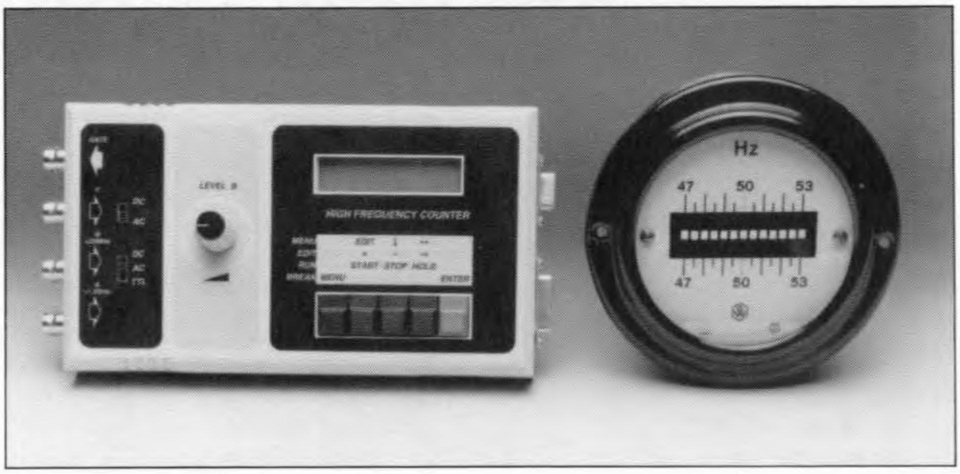
Table 3. Function/code overview.

Main functions	
* First Number	020H
* Measurement function	020H
Frequency	021H
1/Frequency	022H
Rev counter	023H
Pulse counter	024H
* Reserved	025H
* "	
* Reserved	028H
Time	029H
Timer	02AH
Pulse generator	02BH
Zero counter	02CH
Manual counter	02DH
Parameter functions	
* Input	02EH
Channel A	02FH
Channel B	030H
Channel C	031H
Channel C high	032H
Channel C low	033H
Channel C high-to-low	034H
Channel C low-to-high	035H
* Ratio	036H
Period duration	037H
Pause period	038H
Pause duration	039H
* Gate time	03AH
Gate time 0.1 sec	03BH
Gate time 1 sec	03CH
Gate time 10 sec	03DH
Gate time 1 min	03EH
Gate time user def.	03FH
Gate time measured	040H
Gate time rising	046H
Preset period duration	047H
Preset pulse/pause	048H
User def. period	049H
User def. duration	04AH
User def. pause	04BH
User def. pulse	04CH
* Number of periods	04DH
* Reserved	04EH
* "	
* Reserved	052H
* Start value	053H
Start value nought	054H
Start value user def.	055H
* Gate	056H
Gate preset	057H
Gate Channel C high	058H
Gate Channel C low	059H
Gate Channel C high-to-low	05AH
Gate Channel C low-to-high	05BH
Gate key pressed	05CH
Gate START-STOP	05DH
* Start of measurement	05EH
Start immediately	05FH
Start on signal	060H
Start on signal A	061H
Start on signal B	062H
Start on channel C high-to-low	063H
Start on channel C low-to-high	064H
Start on START key	065H
* End	066H
End on no. of periods	067H
End on channel C high-to-low	068H
End on channel C low-to-high	069H
End on STOP key	06AH
* Output	06BH
Pulse on start/end	06CH
Pulse on start	06DH
Pulse on end	06EH
Pulse from start to end	06FH
Active when NE0	070H
Active when EQ0	071H
Pulse on EQ0	072H
Count pulse	073H
Counter setting functions	
* Miscellaneous	074H
Measurement order	075H
Continuous	076H
Single	077H
Buzzer	078H
Beep on	079H
Beep off	07AH
Intermediate value	07BH
With interm. result	07CH
Without interm. result	07DH
Period analysis	07EH
Period analysis on	07FH
Period analysis off	080H
Pulse polarity	081H
Pulse polarity pos.	083H
Pulse polarity neg.	084H
Inactive level	085H
Inactive level low	086H
Inactive level high	087H
* Reserved	088H
Remaining functions	
* Input	089H
* Reserved	08AH
* "	
* Reserved	092H
Main Break	093H
Reset	094H
* System	095H
* Reserved	096H
* Reserved	097H
Run command	098H
Buzzer	099H
* Reserved	09AH

* = not significant for PC control

indeed arbitrary. This potential problem is solved by having the computer adjust the gate time such that counter 1 processes a whole number of periods. This rules out errors in the number of pulses counted by counter 1. The timing diagram in Fig. 13 shows what happens. After the gate time T has elapsed, the system keeps counting for a time Δt, so as to include the input signal period that has just started. Unfortunately, the above 'trick' can not be applied to counter 2. That is nothing to worry about, however, provided the counter is fed with a great many pulses. If this is so, the error introduced by the single missing pulse is considerably reduced, as already explained in the section on the classic frequency meter. The number of pulses to be counted by counter 2 depends on the reference frequency (f_{ref}) and the gate time. The reference frequency being fixed (500 kHz in the case of the multifunction frequency meter), it will be obvious that we must maintain a reasonably long gate time (the shortest gate time that can be set on the instrument, 100 µs, is just about acceptable).

Although an error of one pulse is inherent in the operation of counter 2, there is still a means of increasing the accuracy of the measurement. To begin with, we have the computer provide a fixed logic level (for instance, 0) at the input of the counter when the gate time starts. This is achieved with a software-controlled inverter. In this way, we are certain that the first (already running) period of the reference signal is included in the count as long as possible. Also, we can have the computer check the logic level of f_{ref} at the end of the gate time. In fact, this produces an error that is smaller than one pulse. All in all, we can safely assume an error of one pulse for the error calculation. The relative error made by counter 2 is $1/N2$. To ensure the smallest possible relative error, $N2$ must be as large as possible. This can be achieved by making f_{ref} and/or the gate time as large as possible.



Returning to the measured frequency calculation, $f = f_{ref}\,(N1/N2)$, you may spot another source of errors: the reference frequency. The relative error in this frequency is determined by the quartz crystal used to generate the reference clock. The total measurement error thus becomes:

$$\Delta_{rel} = \Delta f_{ref} + 1/N2.$$

To calculate the error, it is easier to write $f_{ref}\,(T + \Delta t)$ instead of N2, because f_{ref} is known, T is set on the instrument, and Δt is negligible at relatively high frequencies, and easily calculated at low frequencies on the basis of the measurement result. In addition, the more extensive notation indicates clearly that the relative accuracy of the measurement depends exclusively on (1) the reference frequency, (2) its stability, and (3) the time reserved for the measurement, instead of on the measured frequency.

So, what does it all do in the case of the instrument described? Assuming a measurement time of 0.1 s and a reference frequency accuracy of, for instance, 100 ppm (0.01%), the relative error is as small as

$$\Delta_{rel} = 0.01\% + 100\%/(500\text{ kHz} \times 0.1\text{ s}) = 0.012\%$$

or 120 ppm. Obviously, the relative error is even smaller if the stability of the reference frequency is better, and the measurement time longer.

The functions indicated in Fig. 13 are not easily found back in the circuit diagram of the instrument (Fig. 2 in part 1). In fact, only the gate signal (on connector K5) and a piece of counter 1 are obvious, the rest is implemented by the hardware contained in the microcontroller. ■

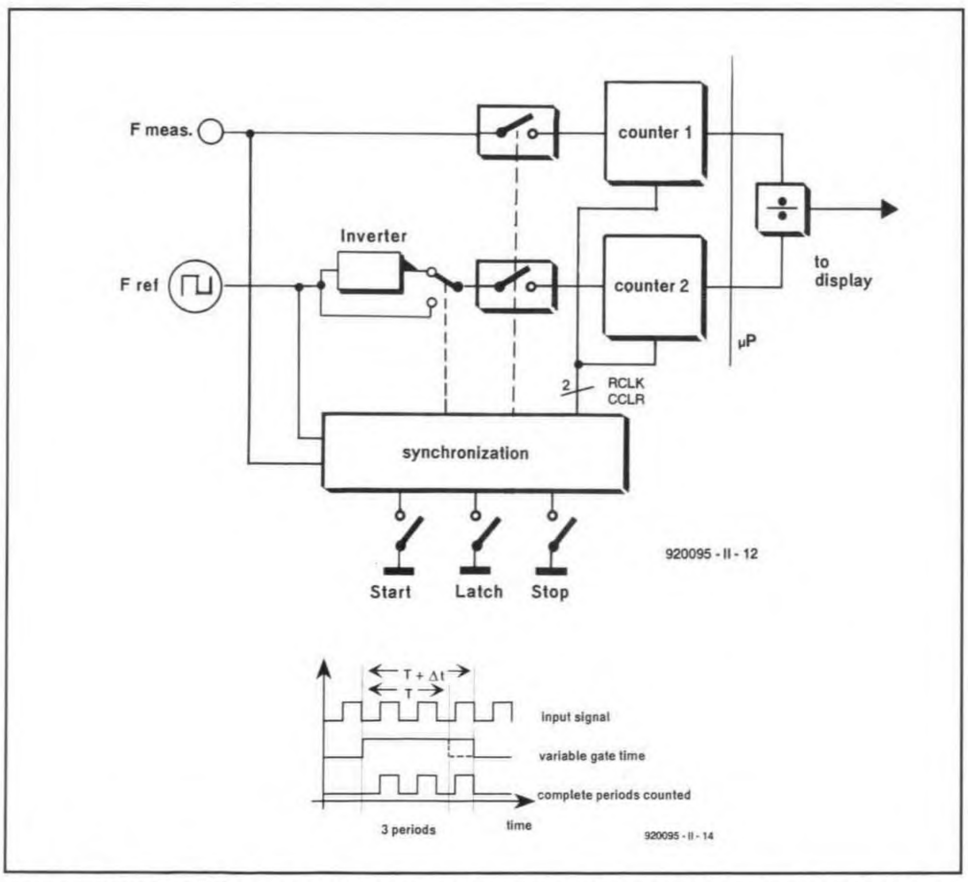


Fig. 13. By virtue of the dual-counter approach, a computer-based frequency meter achieves greater accuracy than a 'classic' design (compare Fig. 12).

References:

1. 'Microprocessor-controlled frequency meter'. *Elektor Electronics* January 1985.

2. 'Multifunction measurement card for PCs' *Elektor Electronics* January and February 1991.