AN1147
APPLICATION NOTE

High Resolution Digital Conversion of an NTC by ST52x301

Authors: A. Cucuccio, G. Rascona'

1. Introduction

In this application note is presented an efficient and flexible method to evaluate the value of a variable resistor in the range 1KΩ ÷ 10KΩ (i.e. it could be an NTC resistance for a thermal regulation control process).

The level of accuracy that can be reached by using a Timer integrated in the ST52x301 micro and an external circuitry in addition to the microcontroller (mirror current circuit) is very high.

The evaluation of the resistance is done through an indirect measurement related to the determination of the charge time (at constant current) of a capacitor included in the mirror current.

The high stability together with the precision of this time interval measurement is strictly related to the features of the ST52x301 on-chip Timer. It is a Timer fully programmable and configurable by the user so that it results easy and flexible to use and, at the same time, allows to reach very high resolution of the evaluated measure.

The indirect methodology used in this application note is only an example of the various applications where the presence of a timer is required to determine the physical quantities involved in a control process algorithm.

This script aims to show to the user a typical application of the Timer and its powerful and easy management in FUZZYSTUDIO™3.0 environment. Moreover, a serial communication between the micro and a PC architecture is developed by providing both the software program and the hardware interface necessary to implement the communication between the two systems.

2. The ST52x301 Timer Peripheral

ST52x301 offers one on-chip Timer peripheral consisting of an 8-bit counter with a 16-bit programmable Prescaler, giving a maximum count of 2^{24}, and control logic that allows the working configuration and the setting of the type of peripheral outputs.

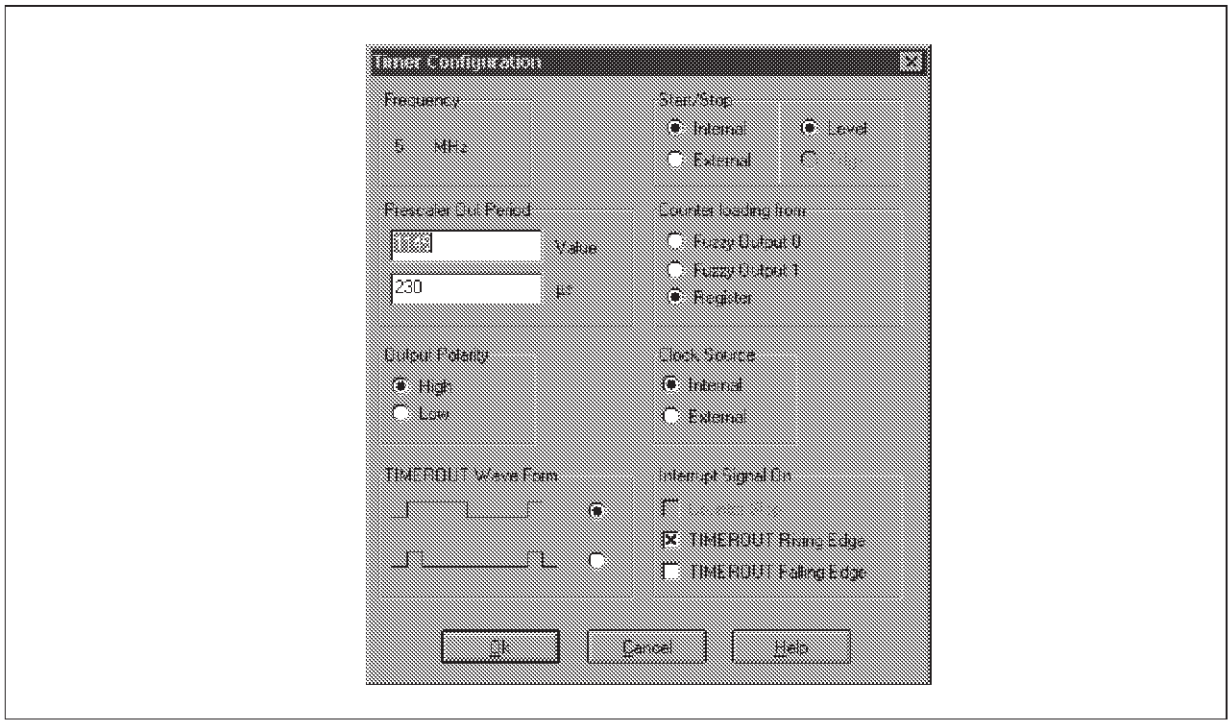
The content of the 8-bit counter can be read/written and is incremented on the rising edge of the 16-bit Prescaler output. Moreover, it can be read under program control at any instant of the counting phase and loaded into a variable. Any value between 0 and FFFFh can be assigned to the Prescaler.

In figure 1 is shown the dialog box of FUZZYSTUDIO™3.0 "Timer Configuration" with the settings that have been used in our application.

A brief description of Timer settings follows:

- **Prescaler Out Period**: the value of the prescaler out period in µs or in the corresponding decimal value (16 bits word);
- **Ouput Polarity**: defines the polarity of the Timer output signal (TIMEROUT);
- **TIMEROUT Wave Form**: this is a signal with frequency equal to the working Timer frequency divided by the starting value of the Prescaler (16 bits) and Counter (8 bits). The Timer output can be either a square wave with duty-cycle 50% or a pulse signal (with pulse duration equal to the Prescaler Ouput signal period; see figure 3);
- **Start/Stop**: the start/stop of the Prescaler counting can be provided from an internal or external source. Moreover the start/stop of the counting depends on either the level or the edge of a TSTART signal.
- **Counter loading from**: selects the source of the Counter value between a location of the Register File and the Fuzzy Core;
- **Clock Source**: selects the source of the working Timer frequency;
- **Interrupt Signal**: the Timer can be programmed to generate an Interrupt request until the end of the count or when there is a falling and/or rising edge in the TIMEROUT signal (see figure 3).

Fig. 1 - Timer Configuration



In figures 2 and 3 the Timer functionalities and the timing of TIMEROUT signal are shown respectively. For more details on the Timer's functions, configuration and interrupts refer to ST52x301 Data Sheet.

Fig. 2 - Timer Features

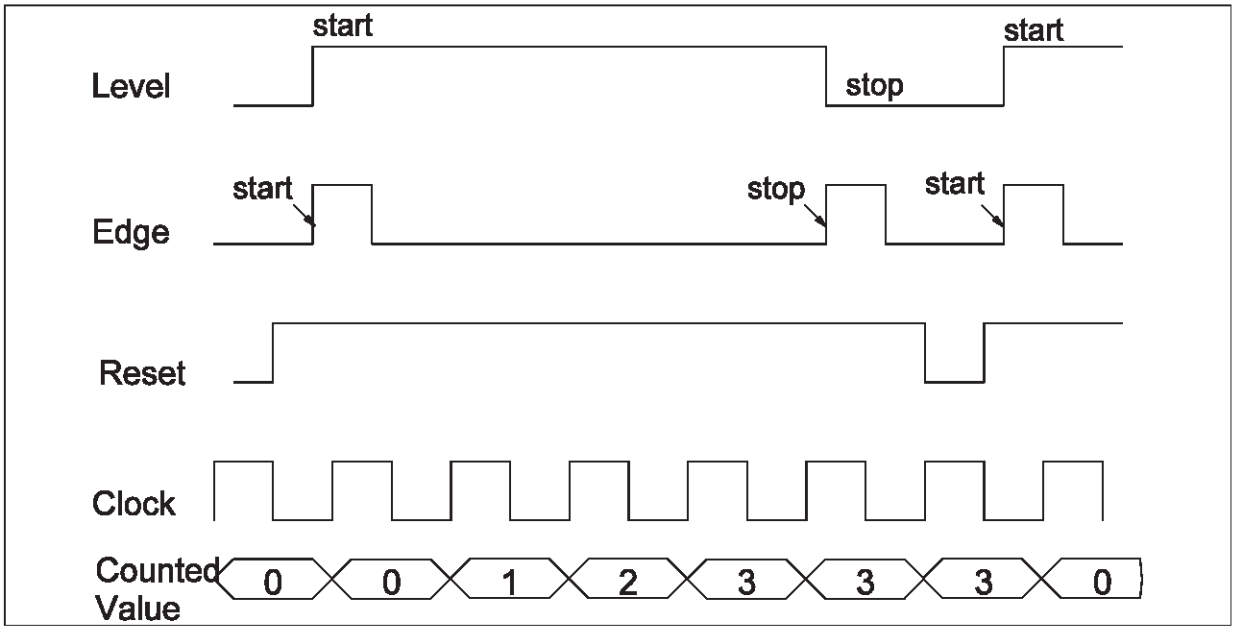
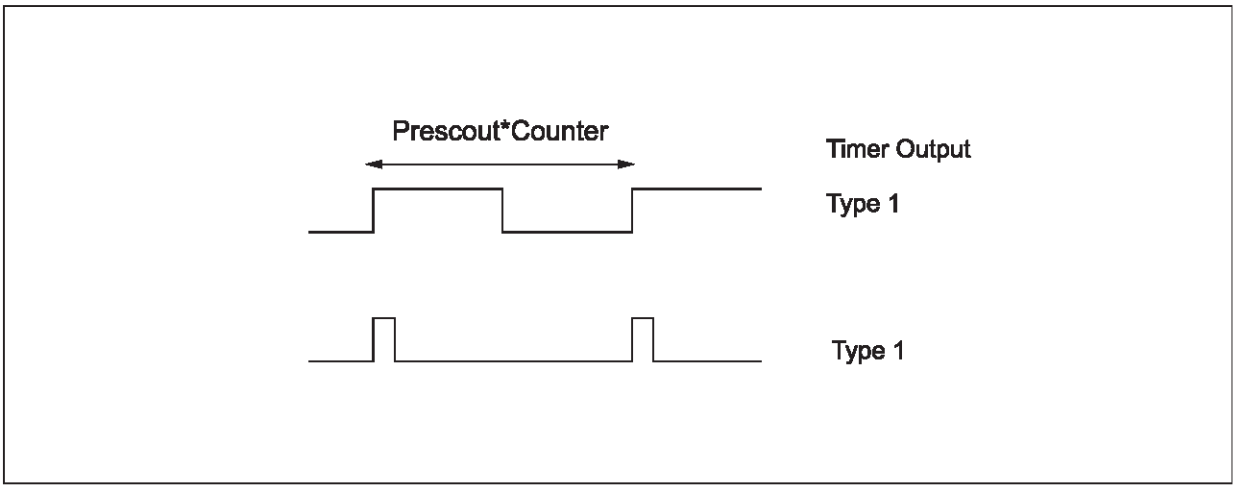


Fig. 3 - ST52x301 TIMEROUT Signal Type



3. General Description

Let us suppose we want to measure the resistance value of a variable resistor (i.e. a trimmer) in the range from 1K up to 10K. Any other range of values can be considered without any significant changes to be made on the methodology described later on.

As stated in the introduction, our aim is to determine a resistance value by measuring a time interval related to it (in our case this relationship will be linear).

The time interval is determined by using the programmable Timer of ST52x301 microcontroller whose key features have been already described previously.

The main principle of the resistance-time conversion is based on determining the necessary time so that a capacitor, with initial voltage equal to zero, is charged up to a fixed voltage threshold value V_{th} (threshold voltage of a Schimitt's trigger making up the input stage of a generic input pin of the parallel port in the microcontroller) .

Thus, if the charging current of the capacitor is kept constant, the relationship that relates our quantities is:

$$V_{th} = I(R)/C \cdot T \qquad (1)$$

where:

- **V_{th}**: is the threshold voltage.
- **I(·)**: is the time-independent current depending on the resistance value. It allows the linear charging of the capacitor. This is accomplished by means of a mirror current circuit linked externally to the micro;
- **R**: is the current value of resistance that determines the corresponding time interval during which the capacitor is charged;
- **C**: is the capacitance that allows to obtain the resistance-time conversion. Its value is determined in order to have a certain time to gt the measure and also it depends from the current intensity flowing through the mirror current. In our case we have chosen a 33 µF capacitor;
- **T**: is the time needed for the capacitor voltage to vary from 0V up to the threshold voltage value (about 1.9V).

We can observe that is just the constant current charging that allows to obtain a very linear relationship between the resistance value and the relative time interval determining a measure of it.

As the current flowing in the capacitor is inversely proportional to resistance, the following relationships are valid:

$$I(R) = K/R \qquad (2)$$

where K is a generic constant, and, from (1):

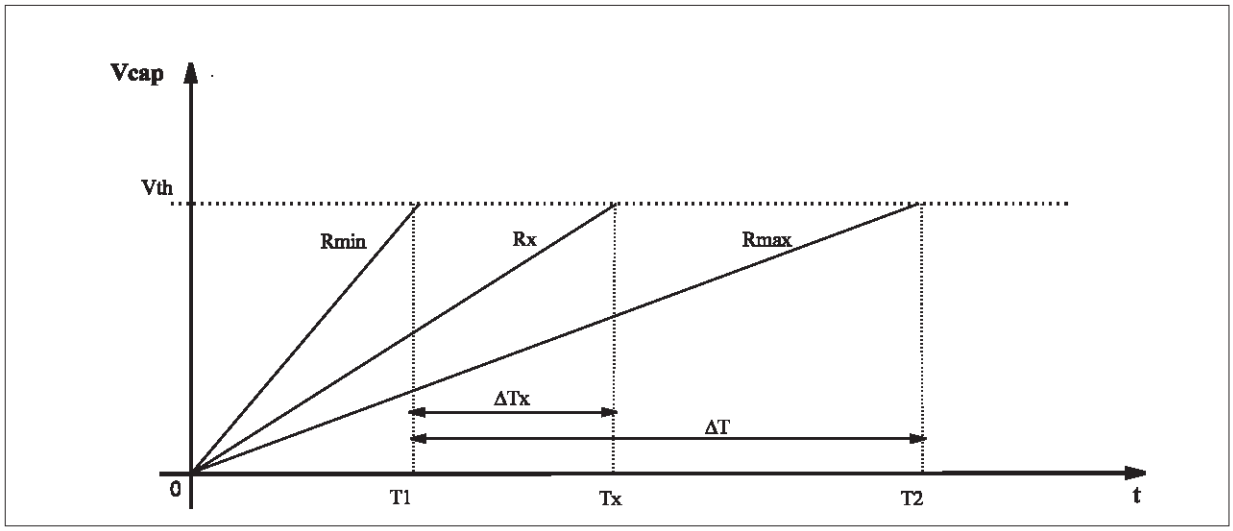
$$T = (V_{th} \cdot C)/K \cdot R \qquad (3)$$

In figure 4, on a time-voltage diagram, are drawn the ideal shapes of the voltage across the capacitor for different resistance values.

In particular, we notice that at the minimum value of resistance R_{min} corresponds the minimum time T_1; similarly at the maximum resistance R_{max} corresponds the biggest charge time T_2.

For values of resistance laying in the range [R_{min} , R_{max}] the voltage is represented by the straight line related to a generic R_x corresponding to a time interval T_x included between the extremes T_1 and T_2.

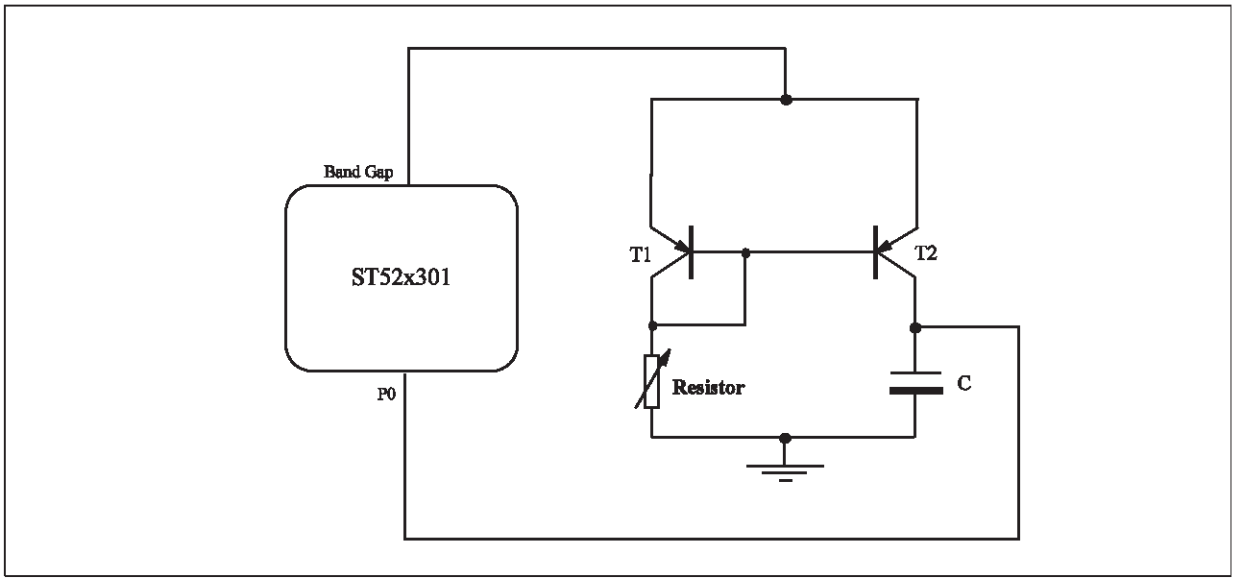
Fig. 4 - Voltages across the capacitor in the charging interval at constant current



4. Link between the ST52x301 Micro and the Mirror Current

As said before, the charging phase of the capacitor is at constant current and this is obtained through an external mirror current circuit. In figure 5 the connections between the micro and the mirror current are shown.

Fig. 5 - Mirror current driven by ST52x301 micro



The mirror current is supplied with the highly-stable bandgap voltage reference provided by our micro. To ensure an optimal performance of the circuit, it should be better to use matched transistor with the same electrical and thermal characteristics.

From the figure, we can note that the key components of our measurement system, the resistor and the capacitor, are connected to the collectors of transistors, and the voltage across the capacitor is detected by pin 0 of parallel port.

Now, let us analyse the functionment of pin 0 (PIN0) that is a bidirectional line. At the beginning of measurement cycle, PIN0 is configured as output by the program and set to logical value 0 (analog system ground): in this way the capacitor is fully discharged. After, the PIN0 is placed as input pin (high impedance); at the same instant the timer starts counting. The voltage across the capacitor is free to grow linearly up to the threshold voltage V_{th} (constant current charging).

The running program loops checking the state of PIN0 that, initially, is at logical 0 but switches to logical 1 after few milliseconds depending on the resistance values. When this commutation occurs, the program interrupts the polling cycle and blocks the timer counting. The total time is stored into two registers (one of which is the Timer counter).Then, the time Offset due to the minimum charge time T_1 has to be subtracted from this 16-bit value giving an 8-bit data representing the result of A/D conversion.

5. Measure Resolution

We have to determine an adequate number of bits necessary to represent our measure, once the maximum range of variation of the resistance has been fixed.

It is obvious that a major number of bits implies a greater resolution and, therefore, precision of measurement.

In this application note we will impose a word length so that the resolution is at least 50Ω/LSB (i.e. a variation in the less significant bit implies at the most, a resistance variation of 50Ω).

It is known that:

$$R_{min} = 1K\Omega \quad \text{and} \quad R_{max} = 10K\Omega \quad \text{then} \quad \Delta R = 9000\Omega$$

If we use an 8-bit representation, we will have:

$$\Delta R = 9000\Omega / 2^8 \cong 35\Omega/LSB$$

With the above considerations we have enough information to set the Timer configuration and to implement the structure of the measurement algorithm.

Referring to figure 4, T_1 represents the smallest time interval when the trimmer has its minimum resistance value (1KΩ); in these conditions our 8-bit word has to be equal to 0; in the same way, for the greater time interval T_2, the word value has to be 255.

From practical measurement, we have obtained the following values (with a 33µF capacitor):

$$T_1 = 107ms \quad \text{and} \quad T_2 = 166ms$$

Then, the effective time interval that must be measured will be:

$$\Delta T = (T_2 - T_1) = 59ms$$

and, using an 8-bit length, we get the smallest appreciable time interval:

$$\delta t = \Delta T / 256 = 230\mu s$$

With this value we set the Prescaler register as we saw in previous paragraphs; instead, with our resolution, the Timer counter will be set at 255.

Summarizing, the 8-bit value will represent the measure proportional to the time interval that in figure 4 we have indicated with ΔT_x.

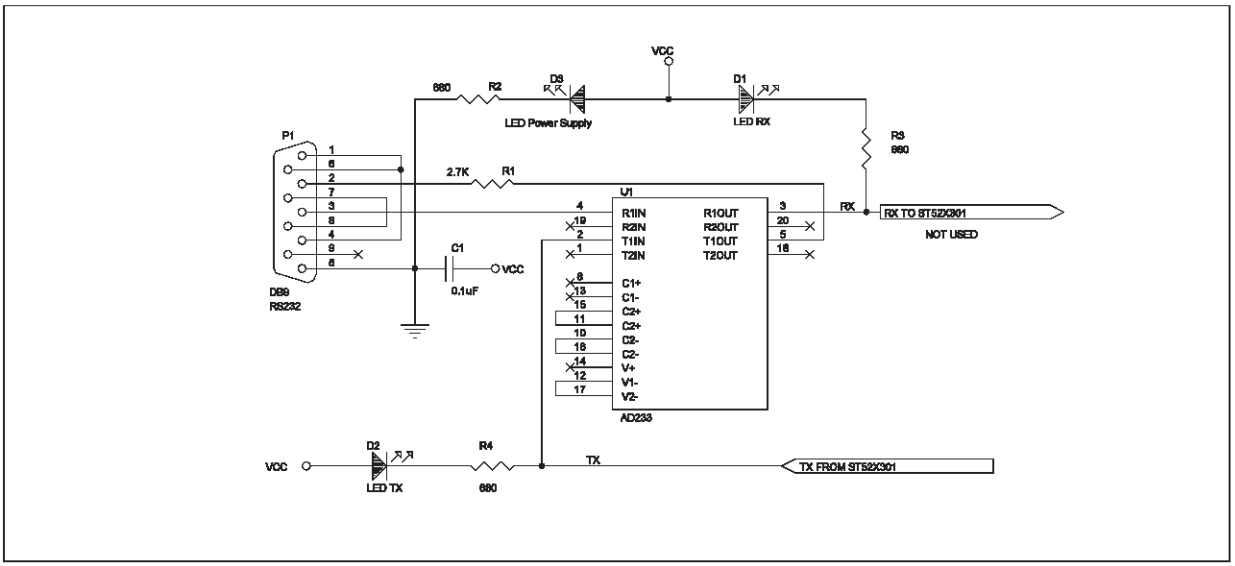
6. Serial Communication Interfacing

The micro ST52x301 contains a Serial Communication Interface (SCI) that provides a general purpose shift register peripheral, allowing to link several widely distributed MCUs, through their SCI subsystem (PCs too).

The SCI gives a serial interface providing communication with common baud rates, up to 38400 Hz, and flexible character format. For more information on SCI refer to ST52x301 Data Sheet.

In figure 6 is provided an interface circuit for serial data transmission from ST52x301 and a PC (through one its COM: port).

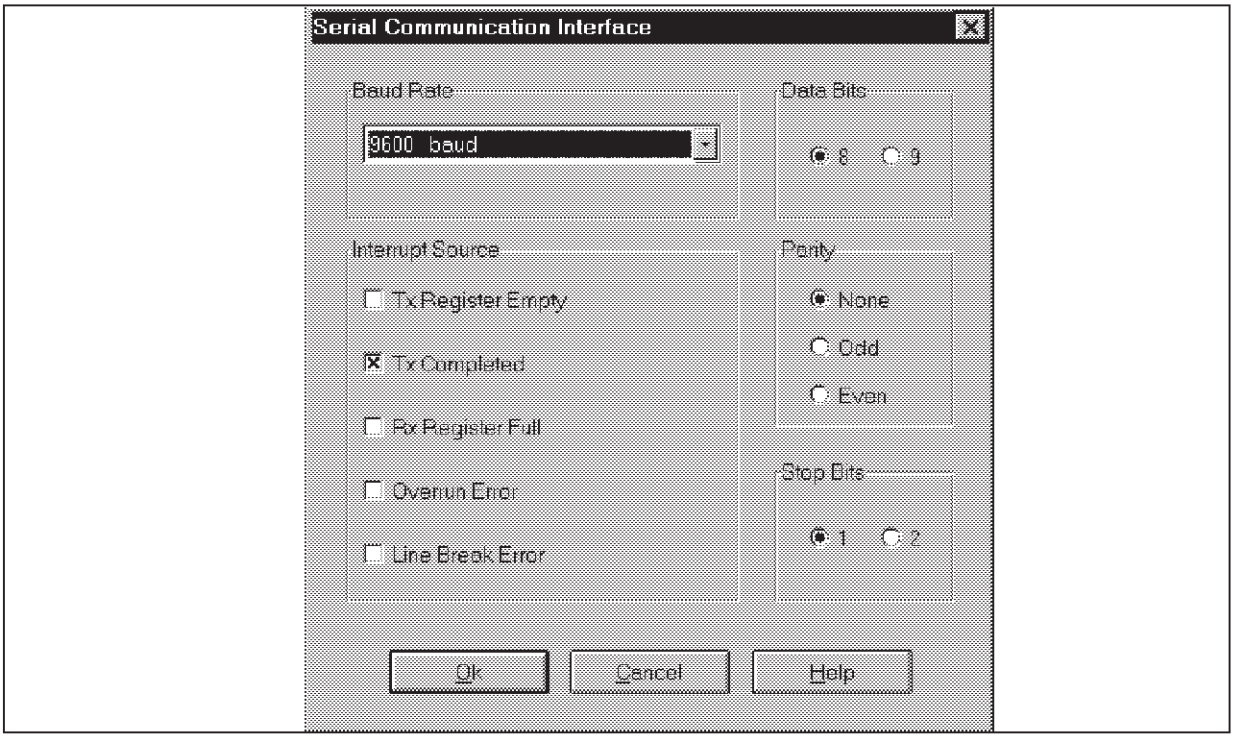
Fig. 6 - Schematic of rs232 interface



In our application, for each measurement cycle, the software will transmit the byte containing the evaluated measure. This allows real time processing on evaluated measures or post-processing if you decide to store the information in a media.

In figure 7 is shown the dialog box from FUZZYSTUDIO™3.0 for SCI configuration with the settings used in our application.

Fig.7 - SCI configuration



7. Software Description

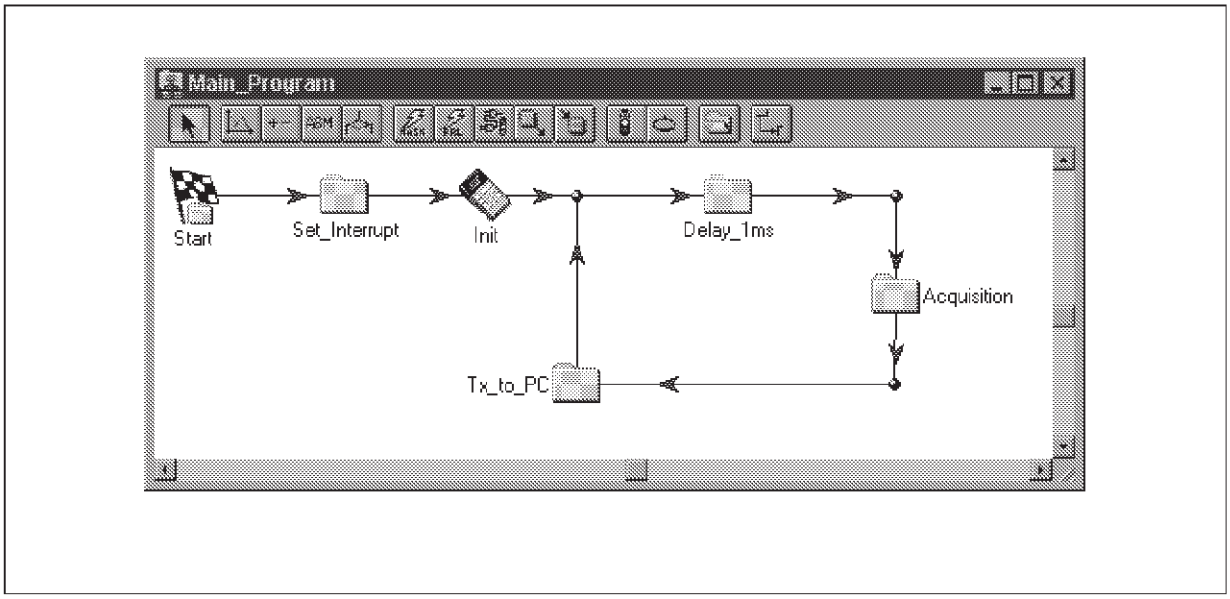
The software running on the micro can be divided into three blocks:

- **Main program**
- **Acquisition Phase**
- **Data Transmission via serial port**

7.1 Main Program

The structure of the main program is shown in the figure below:

Fig. 8 - The Main Program



Let us analyze each block:

SET INTERRUPT: Sets the interrupt priorities.

INIT: Sets to zero the two bytes containing the upper and lower part of acquisition performed during the acquisition phase.

DELAY_1ms: A delay cycle between a conversion and the next one.

ACQUISITION: Routine that computes the time interval measure related to resistance value. For more details refer to the Acquisition Phase block.

Tx_to_PC: Block in which the data is sent output via serial port pin TxD. See below for more details.

7.2 Acquisition phase

In figure 9 is shown the block diagram (in FUZZYSTUDIO™3.0 environment) of “Measurement” routine where the effective detection of measure is performed.

Before we begin to analyze this routine, we have to explain how to subtract the offset time due to minimum time T1. To do that, let us see what is the digital value corresponding to this time:

$$T_{min}=107 \text{ ms}$$

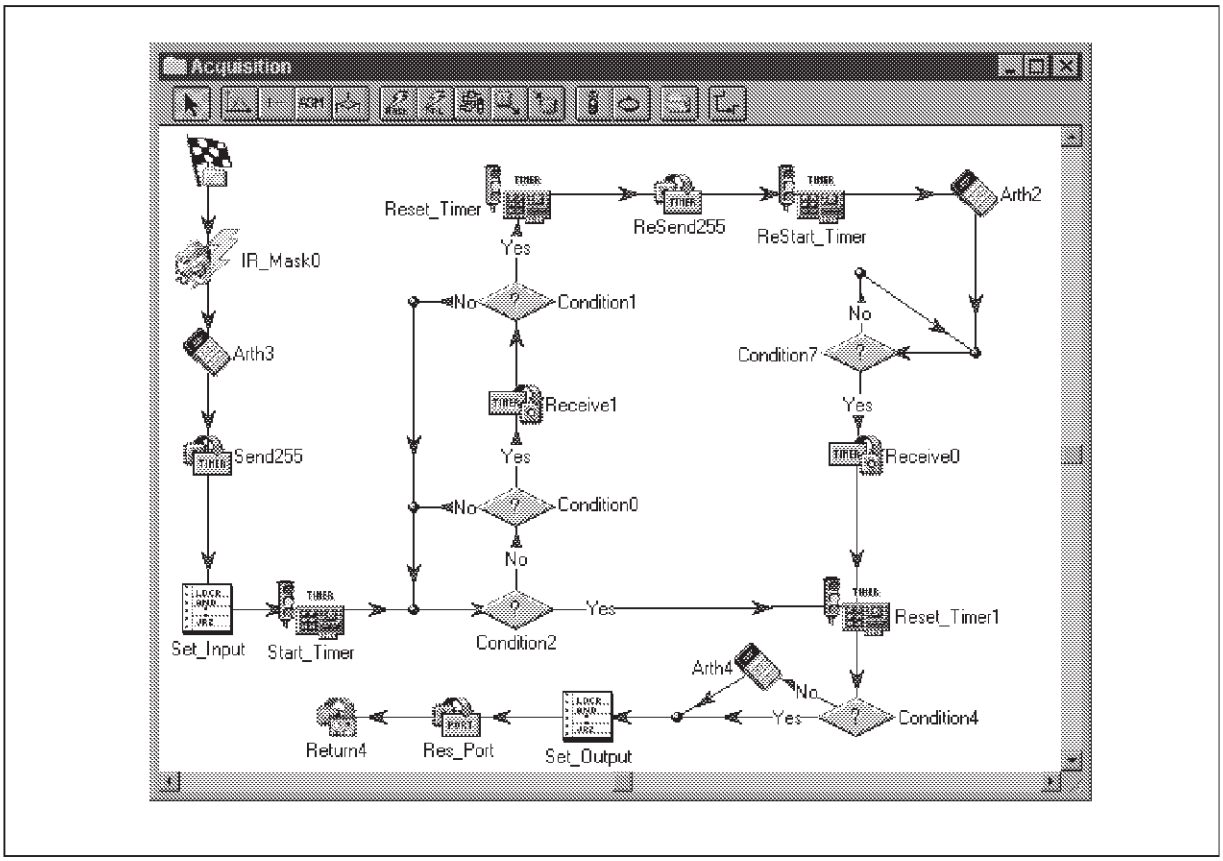
$$\text{Offset} = \frac{T_{min}}{\delta t} = \frac{107ms}{230\,\mu s} = (465)d = (111010001)_b$$

Storing it in two bytes we have:

Upper_byte=1
Lower_byte=209

Then, all we have to do is to start the Timer controlling when this 9-bits value is reached; at this point we have to reset and restart the Timer to count the right time interval we are interested in (the ΔT_x interval shown in picture 4).

Fig. 9 - The measurement routine



Let's analyze the flow of the program:

Set_input: this instruction drives Pin0 to be set as input, lets the capacitor start the charging process.

Start_timer: the timer starts its counting, to discard the offset time T_1 as we discussed before (the 3 subsequent conditional blocks).

"**Condition 7"**: Waits for the voltage across the capacitor becomes equal to V_{th}=1.9V; when this is true, the state of input pin 0 switches from digital 0 to 1.

"Receive 0": then the timer is stopped and the Timer counter, containing the measured time, is stores in the variable "Target".

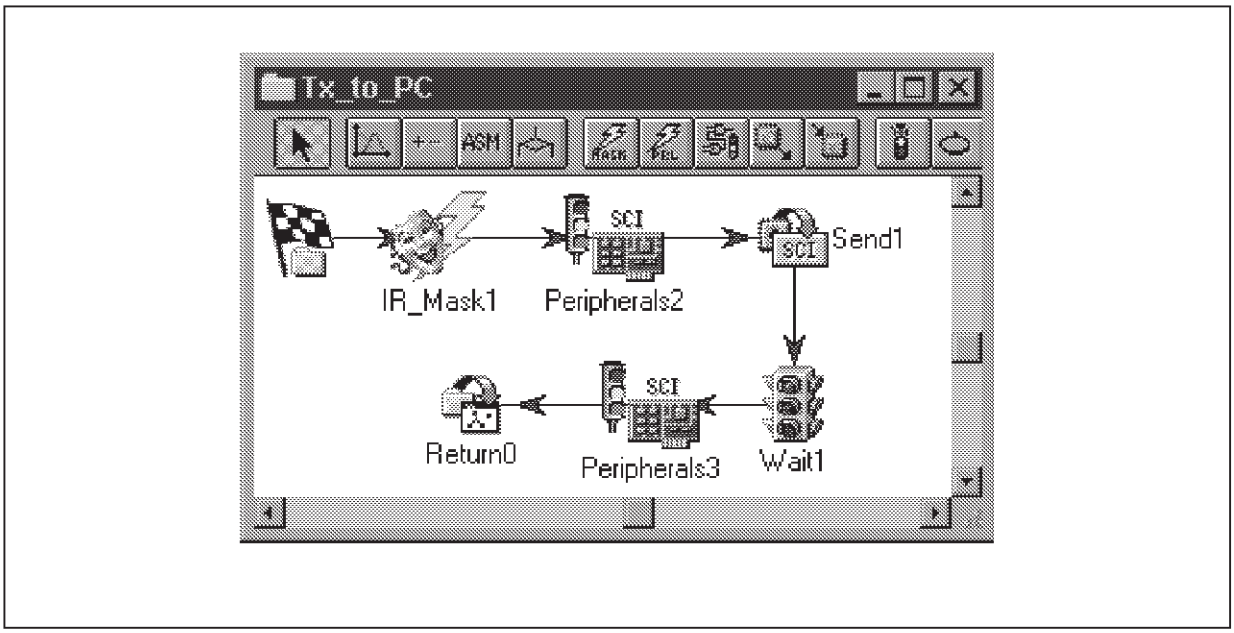
"Set-Output": and Res-Port: PIN0 is set as output at logical level 0 so as to discharge the capacitor towards ground ready for the next acquisition.

7.3 Data transmission via serial port

After the measurement phase, data are sent to serial port. Let us see it in more details (figure 10):

- **IR_mask1:** enables the SCI interrupt.
- **Peripherals2:** starts the transmission phase by the SCI peripheral.
- **Send1:** transmits the variable "res_value" to serial port.
- **Wait1:** the program stops execution until the SCI interrupt is generated; that indicates the data has been correctly transmitted.

Fig 10 - The data transmission to serial port



8. Hardware Description

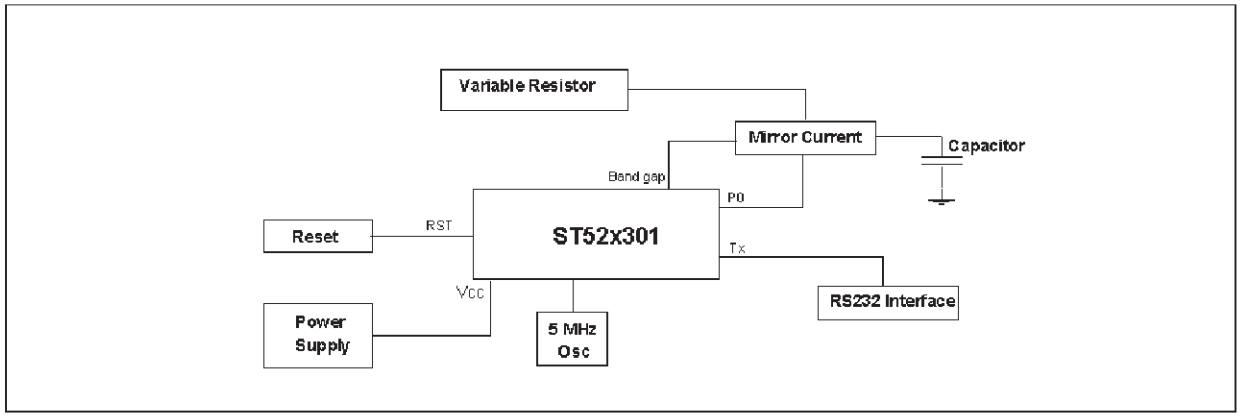
The figure 11 shows the interconnection scheme of all hardware blocks required to implement the application described in the previous paragraphs. It follows that the system hardware can be divided into three main blocks:

- System power supply

- ST52x301

- Mirror current

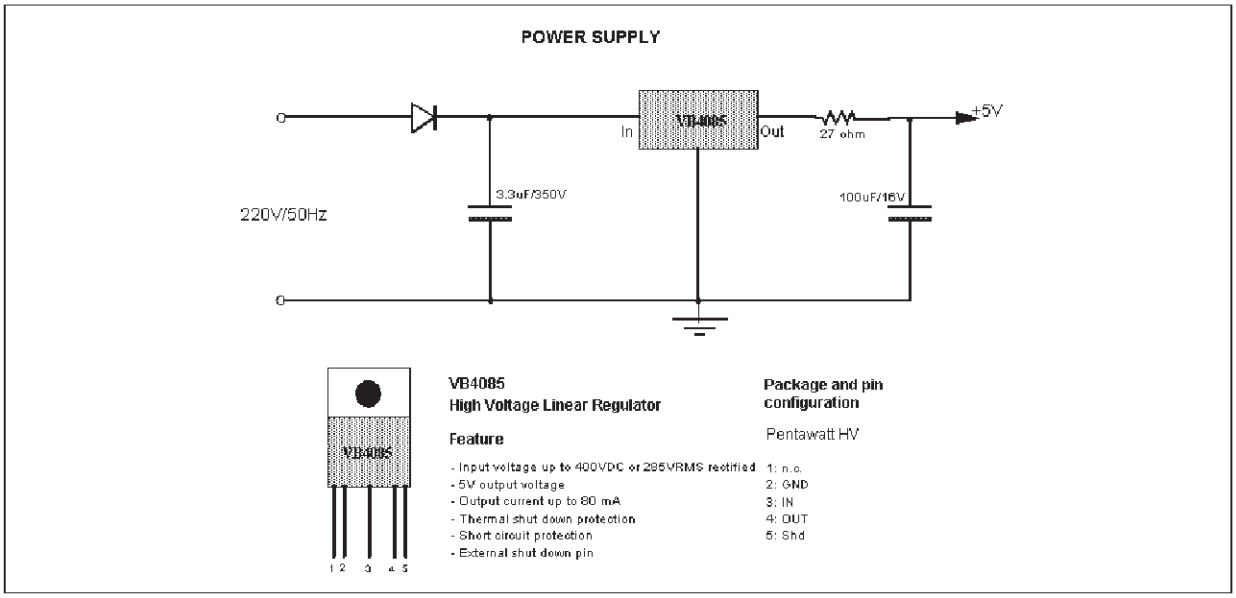
Fig. 11 - The operation scheme of application



8.1 System power supply

Mainly, this part of circuitry is made up from the device VB4085 that allows to have a high-stable regulated output voltage equals to 5V with maximum current up to 80 mA. Further, the DC ouput voltage is filtered by a low-pass filter. With this voltage the digital and analog circuitry of the micro are supplied. Figure 12 shows the power supply circuit used in our application together a brief description of VB4085 features.

Fig. 12 - Power supply section and VB4085 device features



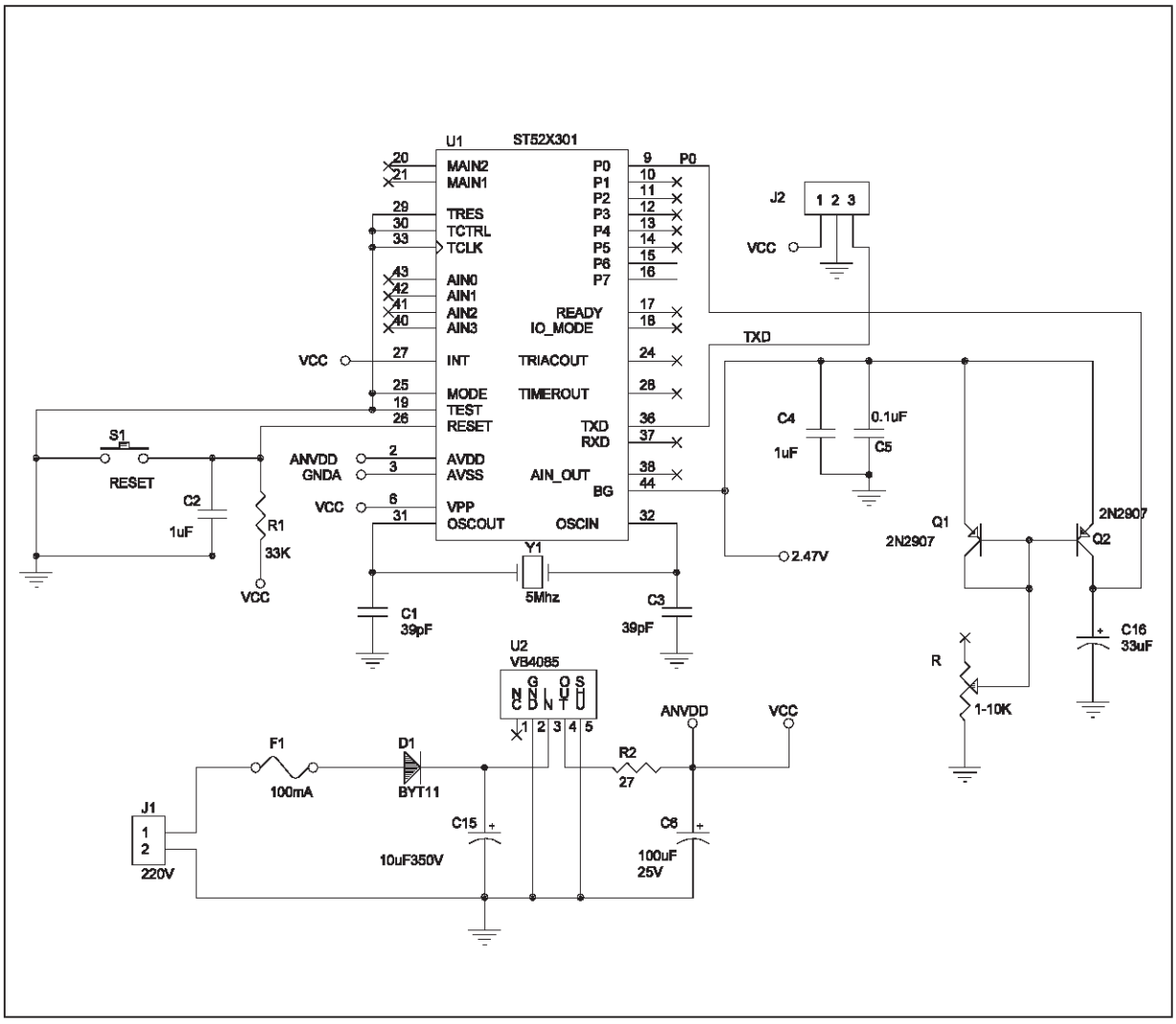
8.2 Microcontroller ST52x301

All main pin connections of the micro are shown. Furthermore, we notice the reset circuit required to begin a new cycle measurement, the 5MHz local oscillator and the connection of the data port pin 0 to the mirror current. The pin TxD is connected to a teeth-connector for an easy access from outside.

8.3 Mirror current

This circuital solution has just been widely described in details above. In figure 13 the schematic of whole application is illustrated. A particular care must be taken by filtering the analog and digital power supplies. This is performed connecting some little filtering capacitors (about 0.1µF) between each Vcc and ground Vss (refers to ST52x301 Datasheet).

Fig. 13 - General schematic of application (without filtering capacitors)



9. An Alternative Solution

The mirror current external circuitry has to be used for constant current charging of the capacitor. This performs a linear A/D conversion of the resistance to be measured.

But, if we prefer not to use external circuitry (i.e. for the constraint of transistors matching, or to have a more low-cost application), another solution can be taken into account.

Instead of using a constant current charging process, we can supply the capacitor with a constant voltage (the DC power supply) from an output pin of the Parallel Port set to digital value '1'. Let us see the basic analytical principle and how this works.

During the charge transient, the voltage across a capacitor in a RC-network supplied with a step signal is given from the well-known formula:

$$V_c(t) = V_{cc}\ 1-\ e^{-t/RC}$$

If, during the charge transient, the threshold voltage V_{th} ($\leq V_{cc}$) is reached after a time of T_x seconds we have:

$$V_{th} = V_c(T_x) = V_{cc}(1-\ e^{-Tx/RC})$$

Then, it follows:

$$T_x = -\ RC \ln(1-\ V_{th}/V_{cc}) = kCR$$

where *k* is a constant depending on DC power supply and the device characteristics.

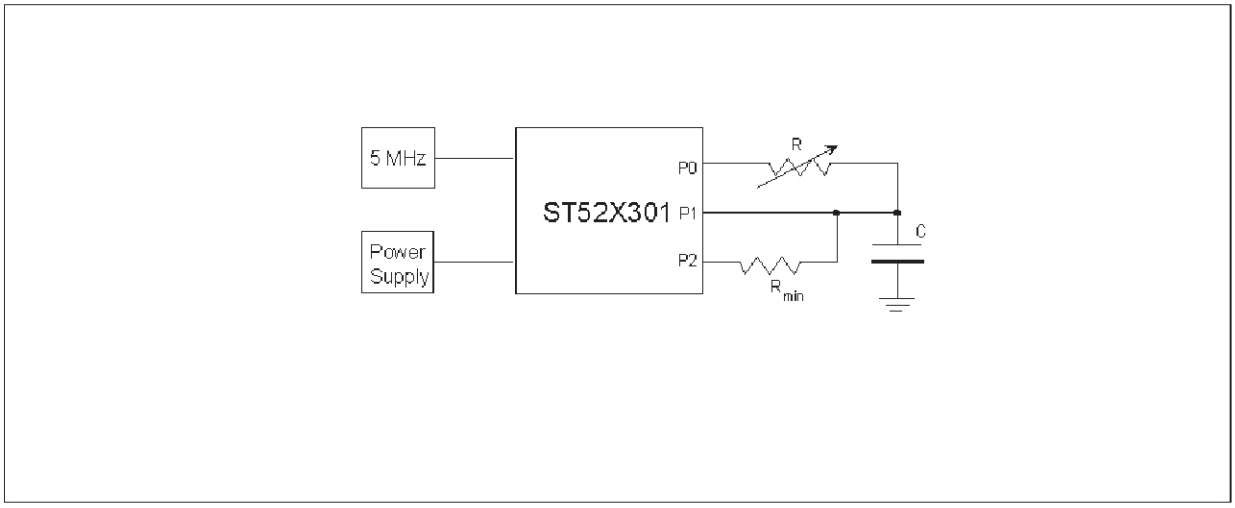
Then, a **linear** relation exists between the time needed to reach the threshold and the resistor R. If we choose a capacitor in such a way that the measurement phase is performed at the beginning of the charge process (in the quasi-linear piece of exponential waveform) then the instant when the threshold is reached will be well-defined (nearly as the mirror current case).

Moreover, with the new solution this is easier to reach because now we are feeding the capacitor with a voltage of 5 V, that is, twice the voltage used in the former case.

9.1 The implementation of A/D conversion

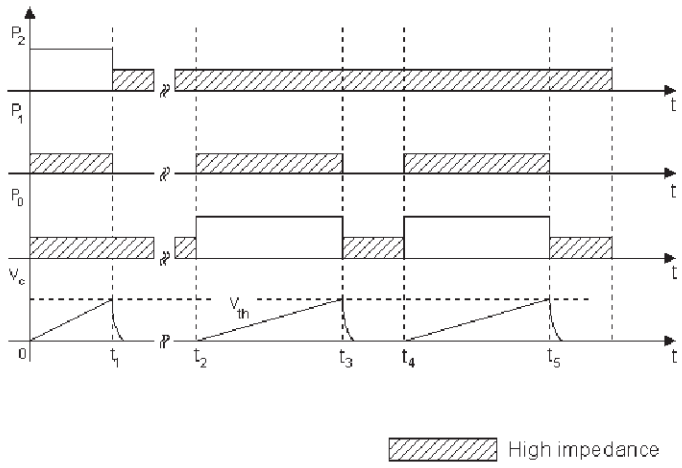
In figure 14 we see the modified circuit driving the variable resistor component for A/D conversion.

Fig. 14 - A/D conversion block diagram



In figure 15 there is the timing diagram showing how the first three pins of Parallel Port (P0, P1, P2) have to be driven by the software to perform the whole phase of measurement.

Fig. 15 - Timing diagram



9.1.1 Pre-processing phase ([0,t_1]):

First of all, we have to find the offset (refer to block 'Init' in software description).

Then, we have to determine the time corresponding to the minimum value of the resistance (in our example 1KΩ). For this purpose we charge the capacitor through the resistor R_{min} rising up the pin P2 of the micro. The time needed to reach the threshold voltage is the wanted offset value.

9.1.2 Measurement phases ([t_2,t_3], [t_4,t_5]):

Normally, in the measurement phase P2 is not used (configured as input/high impedance). For charging the capacitor through the NTC we have to put P0 at logical 1 while starting the Timer. After the voltage has reached the threshold (software polling the pin P1), we stop the Timer, storing its register counter value and, at the same time, configuring P0 as input and P1 as output at zero ground (logical 0) allowing fast discharge of the capacitor towards ground. So, the next measurement phase runs again, after being sure that the capacitor has completely discharged.

10. Determining the Resistance Value of an NTC Sensor

Let us see an immediate application of the method previously described. We want to convert the value of a 10KΩ NTC working in the range temperature of 8, 28°C using an 8-bit resolution.

If the characteristic of the NTC is not-linear we can linearize it putting a 10KΩ resistor in parallel with the sensor.

Experimental measures of the charging times at the minimum and maximum temperatures give for the resistance the following values:

		R_{NTC}(KΩ)	Time (ms) (*)
T(°C)	8	7.2	10.36
	28	4.8	15.44

(*) with a 4.7µF capacitor

Then to convert the range of temperature [8 28°C] in a digital data in the [0, 255] range we have to deal with the following intervals:

$$\Delta T(\ C) = T_{max} - T_{min} = 20\ C$$

$$\Delta t(ms) = t_{max} - t_{min} = 5.08ms$$

With this data we find the value to be loaded in the Prescaler of the Timer:

$$\delta t = \Delta t / 256 = 19.84\mu s$$

We note that with an 8-bit resolution and supposing linear the NTC characteristic, the maximum sensitivity for temperature is:

$$\delta t = \Delta T / 256 = 0.08°C / LSB$$

that is a suitable value in the most common general purpose applications.

Appendix: ST52x301 Assembler code

In this appendix is shown the source program written in assembler language. It refers to the mirror current solution (refer to chapter 8) and it originates directly from the assembler generated with FUZZYSTUDIO™3.0 tool.

```
;       Compile time:     Wed Jul 14 15:02:13 1999
;       Device type:      ST52x301
;       Compiler version: 01.02 (06.11.98)

        Irq             3     Timer_Interrupt
        irq             4     Triac_Interrupt
        Irq             1     AD_Interrupt
        irq             2     SCI_Interrupt
        irq             0     External_Interrupt
                        stop
@@WCLStart@@:
        ldcf            0       0
        ldcf            1       8
        ldcf            2       2
        ldcf            3       128
        ldcf            4       125
        ldcf            5       4
        ldcf            6       40
        ldcf            7       12
        ldcf            8       0
        ldcf            9       0
        ldcf            10      40
        ldcf            11      128
        ldcf            12      66
        ldcf            13      88
        ldcf            14      0
        ldcf            15      228
Start:
Set_Interrupt:
IR_Priority1:
        ldcf            15      198
IR_Mask4:
        ldcf            14      4
Return7:
@@00000:
Init:
        ldrc            13      0
        Ldrc            15      0
```

```
Delay_1ms:
@@00001:

Acquisition:
IR_Mask0:
        ldcf        14      8
Arth3:
        ldrc        12      0
        ldrc        11      0
        ldrc        15      0
Send255:
        mdgi
        ldrc        0       255
        ldpr        0       0
        megi
Set_Input:
        ldcf        0       0
Start_Timer:
        ldcf        6       41
        ldcf        6       43
Condition2:
        mdgi
        ldrc        0       1
        ldri        1       6
        and         0       1
        megi
        jpnz        @@00004
        jp          @@00003
@@00004:
        jp          Reset_Timer1
        jp          @@00005
@@00003:
        jp          Condition0
@@00005:
Reset_Timer1:
        ldcf        6       40
Condition4:
        mdgi
        ldrc        0       1
        sub         0       12
        megi
        pnz         @@00006
```

```
@@00007:
        jp          Set_Output
        jp          @@00008
@@00006:
        jp          Arth4
@@00008:
Set_Output:
        ldcf        0       1
Res_Port:
        mdgi
        ldrc        0       0
        ldpr        2       0
        megi
Return4:
        jp          @@00002
Condition0:
        mdgi
        ldrc        0       2
        sub         0       15
        megi
        jpnz        @@00009
@@00010:
        jp          Receive1
        jp          @@00011
@@00009:
        jp          Condition2
@@00011:
Receive1:
        ldri        13      4
Condition1:
        mdgi
        ldrc        0       209
        sub         0       13
        megi
        pnz         @@00012
@@00013:
        jp          Reset_Timer
        jp          @@00014
@@00012:
        jp          Condition2
@@00014:
Reset_Timer:
        ldcf        6       40
```

```
ReSend255:
        mdgi
        ldrc        0       255
        ldpr        0       0
        megi
ReStart_Timer:
        ldcf        6       41
        ldcf        6       43
Arth2:
        ldrc        12      1
        ldrc        15      0
Condition7:
        mdgi
        ldrc        0       1
        ldri        1       6
        and         0       1
        megi
        jpnz        @@00016
        jp          @@00015
@@00016:
        jp          Receive0
        jp          @@00017
@@00015:
        jp          Condition7
@@00017:
Arth4:
        ldrc        11      0
        jp          Set_Output
Receive0:
        ldri        11      4
        jp          Reset_Timer1
@@00002:
Tx_to_PC:
IR_Mask1:
        ldcf        14      4
Peripherals2:
        ldcf        3       129
Send1:
        stx         11
Wait1:
        waiti
Peripherals3:
        ldcf        3       128
```

```
Return0:
@@00018:
        jp          Delay_1ms
External_Interrupt:
IRET1:
        reti
AD_Interrupt:
IRET4:
        reti
SCI_Interrupt:
IRET2:
        reti
Timer_Interrupt:
Arth0:
        mdgi
        ldrc        0       1
        add         15      0
        megi
IRET0:
        reti
Triac_Interrupt:
IRET3:
        reti
        stop
```