IMPROVING PROCESS CONTROL APPLICATIONS BY USING IP COMMUNICATIONS

Eugen COCA
University "Stefan cel Mare" Suceava
Faculty of Electrical Engineering
13, Universitatii, 720229 - SUCEAVA, ROMANIA
Tel: +40-230-522-978, fax: +40-230-520-277
ecoca@eed.usv.ro

Abstract. *This article presents an implementation of a process control application including a temperature monitoring and control device that may be integrated in more complex process control automations. By using Internet communication and both a web enabled and a local user interface, the device designed and implemented using low-cost embedded microcontrollers offer the possibility to setup all the process parameters in a convenient way. By using standard Ethernet IP network communications the user has the chance to remote control all the process parameters, all data being available in real time to the management unit. The application tested and presented here is very easy to integrate in any existing control system; the range of parameters to control may be extended only by changing the input conditioning process interface components.*

Keywords: *process, control, temperature, communications, interface, Internet*

Introduction

Process control applications include many parameter measurements. One of the common parameter to be determined is the temperature. Low cost microcontrollers are widely used in common applications, because the sensors used are converting the temperature to a voltage or a current by a non-linear equation. Due to high response speed, the computation needs are increasing at this level and the microcontroller is the only suitable solution. By integrating a web enabled microcontroller at this level, we may obtain a low cost Internet controlled device. In this article we present a possible implementation of a temperature control application witch could be used stand alone or integrated in a large process control environment. The microcontroller used for the real-life application is PIC16F876, but the applications may be developed with any low cost microcontroller (i.e. 80Cxxx family). If an analog to digital converter is not present on the chip, the developer must provide an external 12 bit converter. The Ethernet communication uses an embedded microcontroller from the SitePlayer™ [1, 2] family. With few external components and software instructions, we may connect this device to a standard Ethernet port.

Process interface and block diagram

The hardware block diagram is presented in Figure 1. Temperature sensors are connected to the PIC microcontroller PORTA via an interface witch made filtering and signal conditioning [3]. The Local Interface & Display and Control Unit display the temperature values, a history of all measurements and serves as a user to machine interface. All the adjustments made over the keys could be made from web applications as well. The embedded web microcontroller is plugged in an Ethernet port and serves as a HTTP web server.

Temperature sensor interface

Temperature measurement uses LM135, a high precision integrated circuit sensor. Operating as a 2-terminal Zener diode, the LM135 has a breakdown voltage directly proportional to the absolute temperature, with +10 mV/°K and a highly linear output characteristic. The maximum error over a 100 °C measurement interval, when calibrated at 25°C, is no more than 1 percent. Having a dynamic impedance as low as 0.5 - 0.6Ω, LM135 is the best choice for most of temperature control applications [4, 5].

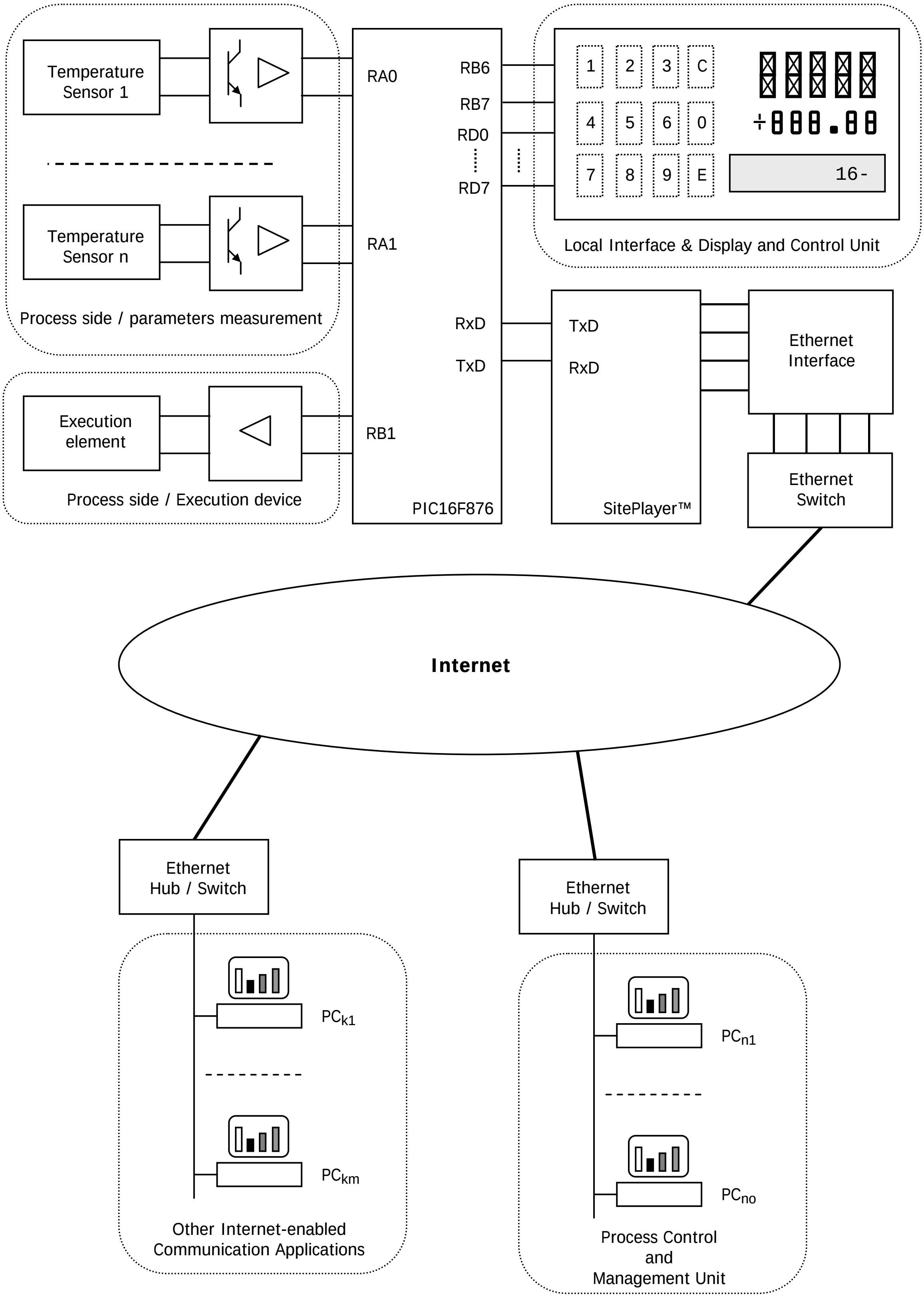

Figure 1. Block diagram of the process control interface with Internet enabled communication

The signal from the sensor is filtered and applied to the microcontroller on RA0 pin, software configured as an analog input. Signals on RA2 and RA3 pins are used as a reference. The second channel is connected to the RA1 analog input. Due to analog to digital signal conversion, we may obtain digital values from 000H to 03Fh, corresponding to the full scale sensor temperature. In order to obtain the correct representation of the real temperature, meaning to decode the dependence between the sensor output voltage and the real temperature, we implemented an assembler routine to make the calculations. This routine consists of a 24 and 16 bit multiplication routine, a 32/16 bit division and a 24 bit subtraction one. The result will be a decimal number, given by the equation:

$$\mathrm{T} = \frac{(\mathrm{V_{in}} - 1\mathrm{C8h}).0\mathrm{C8h}}{199\mathrm{h}} - 032\mathrm{h}\ [°\mathrm{C}] \qquad (1)$$

where T represents the temperature as displayed by the microcontroller at the corresponding input voltage. $\mathrm{V_{in}}$ is the numerical representation of the input voltage on the RA0 analog input. The computation accuracy is highly influenced by the division operation result and this is the reason for we keep the whole division results into two registers. The temperature value is stored in four registers: the first two representing the integer value and two for the fractional one. One bit is dedicated for the sign.

The software interface offers the chance to change the temperatures t_1 and t_2 (Figure 2) for the best response from the device. There is a minimum value for the difference between t_2 and t_1, preset to 0.2 degrees.

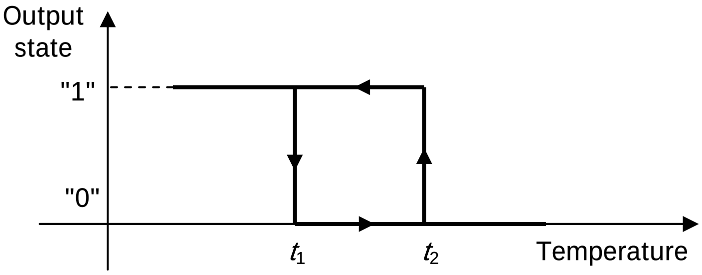

Figure 2. Programmable temperature characteristic

Internet interface

As said in the introduction, the main advantage of this configuration, where we combine both a local and a remote control for the application, is the opportunity offered to the process control application designer to integrate all sensors in unique remote control software. If the idea is used to control only the temperature in a remote location, we may communicate with the device and give commands to the execution unit (a fan, a heating or a cooling unit).

The communication security used by the web microcontroller is a password based one. If the needs are far enough, we may implement an IP address ACL - access control list (on the switch) or we may set up a more powerful security system on the central server. The software application may read all the parameters in real time, display the instantaneous values and keep a local event history. Is a failure is detected in the remote system, the supervisor could be announced via Email, SMS or paging messages.

Timer1 is used to generate fixed-time interval interrupts for temperature measurement. To overcome false alarms, we used a counter and only after six out-of-range temperature values, the execution element is triggered. The used may change the working parameters via the local keyboard or the web-server interface. If a parameter change is detected, no alarm will be triggered and the measurement subroutine will be called from the entry point. For future analysis and decision control, a history of temperature, alarms, parameters and user request is saved on the non-volatile microcontroller memory. The history may be downloaded, archived and cleared via the web interface. An external watch dog circuit is provided.

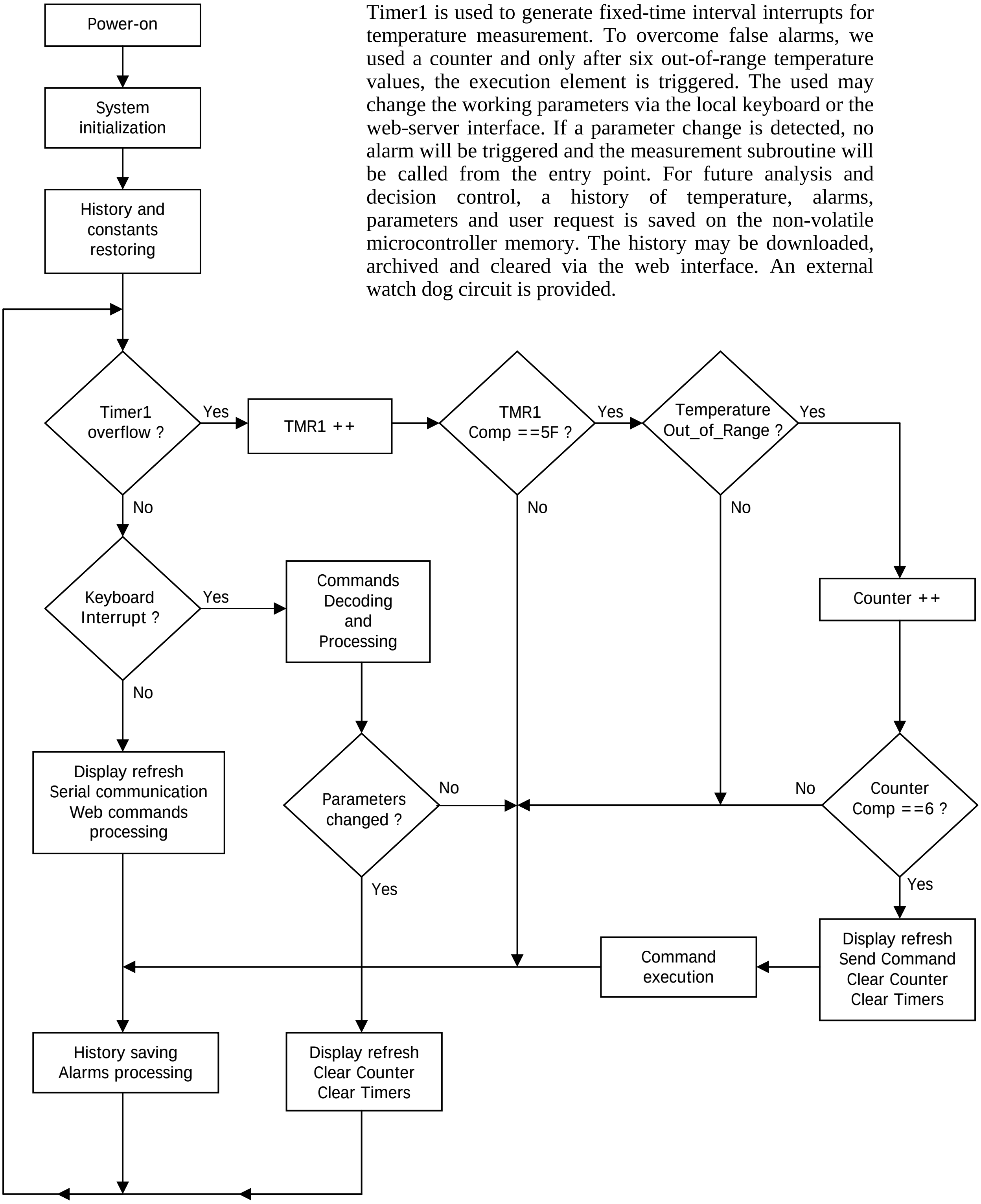
Power-on
System initialization
History and constants restoring
Timer1 overflow ?
Yes
No
TMR1 ++
TMR1 Comp ==5F ?
Yes
No
Temperature Out_of_Range ?
Yes
No
Counter ++
Keyboard Interrupt ?
Yes
No
Commands Decoding and Processing
Parameters changed ?
No
Yes
Counter Comp ==6 ?
No
Yes
Display refresh Serial communication Web commands processing
Command execution
Display refresh Send Command Clear Counter Clear Timers
History saving Alarms processing
Display refresh Clear Counter Clear Timers

Figure 3. Software flow diagram

Software control

As shown in Figure 3, the main part of the software is dedicated to the real time events. After the circuit power on, all the constants and history are restored from the EEPROM memory. Sensor reading is done at regular time intervals, as preprogrammed or introduced by the user via the local interface. If the temperature is in the interval t_1 - t_2 no action should be taken. If the temperature is out of this interval, an alarm LED will signal on the local panel and a corresponding event will be displayed and recorded at the remote Internet location. The execution unit is triggered only if the device was programmed to do so.

The maximum priority is reserved to the calculation routine, the key inputs are read after all sensors are read. The remote display is refreshed automatically, the same value will be both on the local and the remote unit. Timer1 is used to generate fixed-time interval interrupts for temperature measurement. To overcome false alarms, we used a counter and only after six out-of-range temperature values, the execution element is triggered. The used may change the working parameters via the local keyboard or the web-server interface.

If a parameter change is detected, no alarm will be triggered and the measurement subroutine will be called from the entry point. For future analysis and decision control, a history of temperature, alarms, parameters and user request is saved on the non-volatile microcontroller memory. The history may be downloaded, archived and cleared via the web interface. An external watch dog circuit is provided, to overcome a possible software error. After a lack of response the circuit will make hardware reset for the microcontrollers.

Conclusions

The process control applications in use today are local oriented, meaning a remote user could not watch, change or take actions over the process, without going on site. There are many situations where a user should know as many parameters from the process as possible. We could mention here the power supply unit from mobile phone base stations, where the remote operator must know 8 to 10 process parameters to take the correct decision before sending the maintenance team. By web enabling the control unit, the whole application will become more reliable and cost effective. The application presented here may be easily upgraded to 8 or 16 inputs, by adding other microcontrollers or by multiplexing the analog inputs. All circuit components are easily to integrate on a custom device, reducing the costs at a minimum. Further developments are studied: a possible application of this device is on old automation in power stations, where we could control the process without buying new relays and signaling devices.

References

[1] *** (2000) *SitePlayer™ SP 1 Embedded Ethernet Web Server Coprocessor Module*, NetMedia Inc.

[2] *** (2001) *SitePlayer™ SPK1 Web Server Coprocessor Developer Kit*, NetMedia Inc.

[3] *** (1997) *PICmicro Mid-Range MCU Family Reference*, MicroChip.

[4] *** (2000) *LM135/LM235/LM335 Precision temperature sensors*, National Semiconductor

[5] *** (2003) *LM135/LM235 Precision Temperature sensors*, ST Microelectronics.