2005 WSEAS Int. Conf. on DYNAMICAL SYSTEMS and CONTROL, Venice, Italy, November 2-4, 2005 (pp154-157)

Temperature Logger for Testing and Tuning Control Algorithms

DORIN PETREUS, ZOLTAN JUHOS, ALIN CRISTEA
Electronics and Telecommunications Faculty, Applied Electronics Department
Technical University of Cluj-Napoca
26-28, George Baritiu Street, 400027, Cluj-Napoca
ROMANIA
http://www.ael.utcluj.ro

Abstract: - In this paper we propose the implementation of a temperature logger that will measure the ambient temperature according to a certain algorithm and will store the measurement results along with the timestamp of the measurement in a non-volatile memory for later retrieval and analysis. The non-volatile memory is a 32Kbytes EEPROM, and each stored sample is 8 bytes long, resulting in a maximum capacity of 4096 samples. The EEPROM is addressed as an endless circular buffer, so that if the maximum capacity is exceeded, the logger will overwrite the oldest samples from the memory with the latest ones.

Key-Words: Logger, Log-interval, Temperature Sample

1 Introduction

When tuning temperature control algorithms the first problem that arises is the monitoring of the response with the current settings.

As a visual inspection isn't suitable in most of the cases, a logger has to be used that will read a temperature sensor according to a certain algorithm and store the results of the reading in a non-volatile memory.

There are several techniques used for logging:

- rate monotonic logging, the samples are taken at regular intervals from the moment the logger is started until it is stopped;
- bounded, rate monotonic logging, the samples are taken at regular intervals until the result reaches a boundary of a predefined range;
- value sensitive logging, the samples are taken only when there is a significant shifting in the result.

2 Design of the logger

The design of the logger is split up in three major parts: the hardware, the firmware and the PC application.

2.1 The hardware

The logger is based on a C508 microcontroller from Infineon. The microcontroller's instruction set is 8051 compatible, but its architecture is enhanced, executing a machine cycle in 3 oscillator periods instead of 12 like in the case of the classical 8051 core.

For temperature sensing we used the LM75 sensor that can measure the temperature with a precision of 0.5°C and connects to the microcontroller by the means of an I2C bus. I2C is a synchronous serial bus that features two signals: the serial clock (SCL) and the serial data (SDA). All the devices connected to this bus will have an open collector or an open drain topology, and unique address. From the 7 bit I2C address, in the case of LM75 the least significant 3 bits are available for external configuration. These pins are connected to the supply voltage, resulting in the address 1001111b. The choice of the temperature sensor was driven by the fact that it is replaceable without the need of recalibration.

To minimize the hardware requirements, the real-time clock and the EEPROM memory used for storage are also I2C interfaced. The real-time clock (PCF8583) and non-volatile memory (24LC256) used to store the log data are also I2C interfaced, minimizing this way the number of I/O lines taken up from the microcontroller.

The real-time clock's functioning is based on a 32768Hz crystal. From the I2C address, the least significant bit is accessible via a pin. We will set the I2C address to 101000b. The PCF8583 has an amount of 256 bytes of memory, from these the first 16 are used to keep the time, date and alarm settings, and the rest of 240 bytes are available as general purpose RAM. Due to the fact the real time clock can function and retain data at voltages as low as 1.0V, we used a 3.6V/60mAh accumulator to keep the clock functioning when the power supply is disconnected. This way we can take on the

advantage of the 240 bytes of general purpose RAM. These locations will be used for storing the settings for the logger, instead of storing it in the EEPROM.

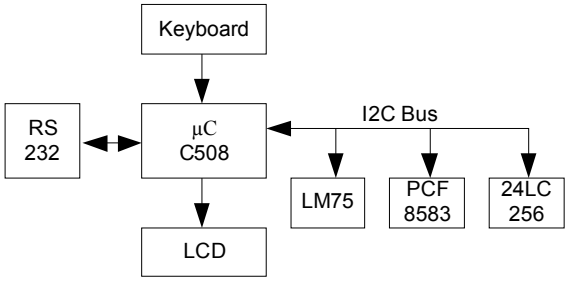

Fig.1 – Block diagram of the temperature logger

The logger's user interface consists of a 4 key-keyboard and a 2 row by 16 characters LCD module. For remotely reading the logged data a computer can be connected to the logger trough the RS232 compatible serial port [1].

2.2 The firmware

The software that runs on the microcontroller is written in C and highly modularized in a hardware-independent manner – except for the low-level routines that directly interact with the hardware – so that they can be ported on other platforms with no or minor modifications.

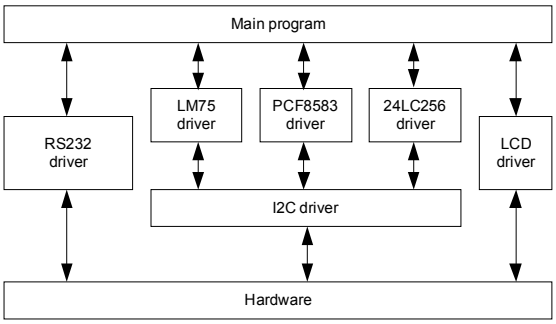

Fig.2 – Software architecture of the logger

The logger has three functioning modes:

-rate monotonic;
-bounded, rate monotonic;
-value sensitive

In the rate monotonic mode the logger will record samples at regular intervals. This mode is proper when a global view of a process is required.

The logging interval can be set by the user to a predefined set of values, starting from 5 seconds up to one hour. After the passing of a log period the logger reads the current temperature and the current time and stores them in the EEPROM.

When the bounded, rate monotonic logging mode is used, beside the logging interval a temperature limit has to be established. After it is started, the logger will record the samples as described in the previous mode and once it reached the temperature limit, it will stop the recording. When this mode is selected, the logging interval can be decreased down to 0.5 seconds.

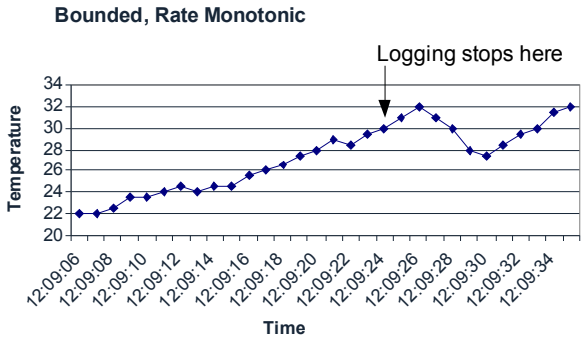

Fig. 3 – Bounded, rate monotonic logging

This mode is especially useful to evaluate the dynamical behavior of a process.

In the value sensitive logging mode, the user will set a temperature limit, so that a temperature variation below it is accepted, resulting a hysteretic behavior. Any variation exceeding the imposed limit will be logged. This way the sample rate will depend only on the variation of the temperature and will be time independent. The first temperature used as reference for the variations will be the one from the moment of starting. In the moment when the sample is recorded, the reference temperature will be replaced by the temperature value from the sample.

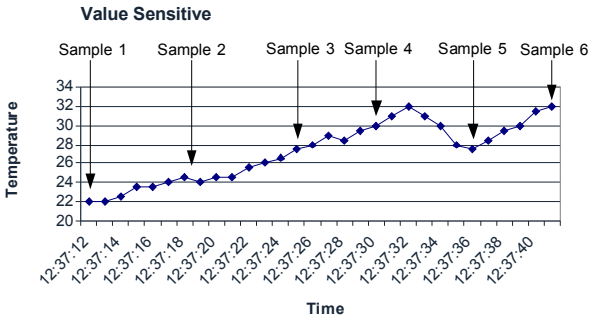

Fig. 4 – Value sensitive logging

This mode is useful when determining process dead-times or monitoring very slowly variable temperatures.

In each logging mode, the sample's structure is the same. It contains the temperature, the date and time, as well as a status byte that will indicate the active logging mode in the moment of recording the sample and weather the current sample is used, unused or deleted. If the current sample is used, it will be displayed when inspecting the log on the local display and it will be downloaded over the serial port. When the sample is marked unused, it will be displayed on the local display, but it won't be downloaded to the PC. If the sample is marked as deleted, the sample won't be displayed at all, thus its value isn't erased from the memory to save the memory endurance.

Temperature High		Temperature Low
Year	Month	Day
Hour		Minute
Second		Status

Table 1 – The structure of a log record

The temperature can be measured in the range -20°C ÷ +125°C, so it has to be stored on two bytes. The date information is stored on two bytes, the hour on three bytes and there is one status byte, that shows that the sample is in use or it is deleted. The rest of the bits from the status byte are reserved for log type to indicate the logging method used to acquire the sample and markers. Each log record has a length of 8 bytes. Having a 32Kbytes EEPROM, that means that we can store up to 4096 samples. If the rate monotonic logging is used with a log period of 5 seconds means that the evolution of the temperature can be recorded for almost 6 hours without loosing a sample.

The log records can be viewed locally on the LCD or can be downloaded to a PC.

The non-volatile EEPROM memory is addressed as an endless circular buffer, so that if the maximum capacity is exceeded the logger overwrites the oldest samples from the memory with the latest ones [2].

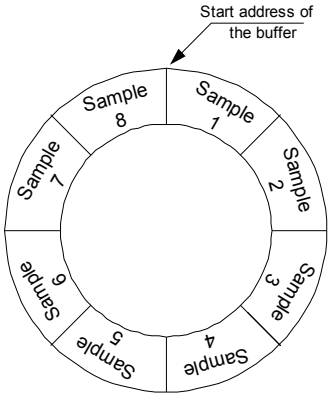

Fig.5 – The classical circular buffer architecture

Usually, a circular buffer (figure 5) is used as a FIFO buffer and has two indexes that indicate where to store the latest data element (usually addressed as "put index") and from where to retrieve the oldest stored data (addressed as "get index"). The length of a circular buffer is preferred to be a power two, so the wrap-around that is made at the end of the physical memory section can be done by simply truncating the upper bits of the address.

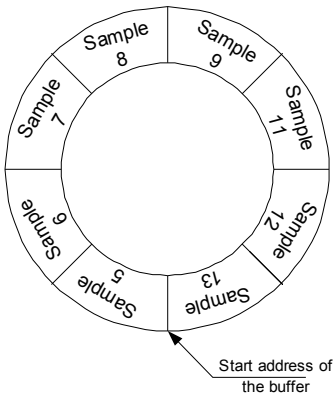

Fig.6 – The endless circular buffer architecture

In the case of our buffer the samples are stored in order and the retrieval can be done in a random fashion. In case of the FIFO circular buffer, the storage of new elements is stopped when the "put index" reaches the value of the "get index", indicating that the buffer is full.

As it can be seen in figure 6, we implemented an "endless" circular buffer. In this case when the buffer reaches its capacity, the oldest sample is overwritten when a new sample arrives.

When the logger is disconnected from the power supply, a track of the last record has to be kept to prevent the overwriting of the previously recorded samples. To do this, after each record we save the address where it was saved. We had two choices for the storage of the address: the EEPROM or the battery backed-up RAM memory from the real time clock circuit. Considering the fact that the locations containing the address of the last record are rewritten very often, storing it in the EEPROM would result in a premature destroying of the memory locations, taking into account the 100,000 erase/write cycles. This isn't a problem when RAM memory is used, so we used the RAM from the real time clock circuit.

The user interface permits the user to set the local time and date, to set the log interval, view the log records or delete the records one by one or all at once. To simplify the user's interaction with the

logger, all the functionality of the user interface is also available through commands issued on the serial port. This way the PC application can do the same settings in an easier and more elegant way.

2.3 The PC application

The PC application that communicates with the user is implemented in C programming language, using National Instrument's LabWindows.

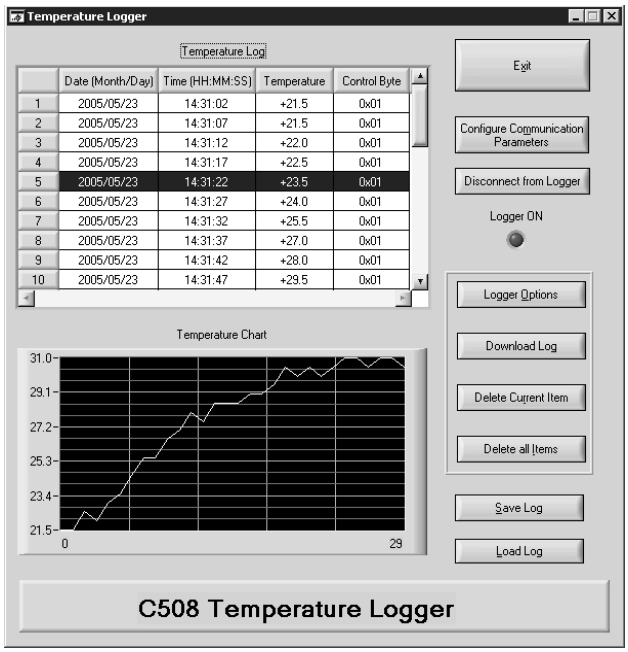

Fig.7 – The PC application's user interface

The PC application's user interface exhibits two types of representations of the recorded data, a spreadsheet representation and a chart representation. The spreadsheet representation allows the user to inspect the log records on a sample-by-sample basis, while the chart representation gives a complete image of the temperature's evolution over the entire log interval. This way the user gets a complete image of the monitored system's behavior.

The application can save the downloaded data in a CSV (Comma Separated Values) file that can be read by most of the common spreadsheet applications or it can load previously saved files.

3 Conclusion

The temperature logger proposed in this paper helps analyzing the efficiency of certain temperature control algorithms in order to fine-tune them specifically for the controlled area. It gives a full image of the process under observation from the moment of start-up, until it is stopped.

The different logging modes can be used to monitor a certain part of the observed process.

Fig.8 – The temperature logger

References:

[1] Cupcea Nicolae, Stefanescu Costin, *Sisteme inteligente de masura si control,* Editura Albastra, 2002

[2] Petreus Dorin, Muntean Gabriel, Juhos Zoltan, Palaghita Niculaie, *Aplicatii cu microcontrolere din familia 8051*, Mediamira, 2005