Alarm/Interlock system

In the UML notation

By Peter Kravtsov (E-mail: PAKravtsov@lbl.gov)

June 2000

1. Requirements Analysis

Alarm/interlock system is a device designed to watch the critical sensors and tak corresponding actions in case of certain fault conditions. The sensors are read by microcontroller with the help of 16-bit ADC. The user can easily prevent any particular alarm event from being processed by the system. The device comes with appropriate PC configuration and monitoring software. We will examine our system as two independent parts whereas once configured, the device itself can work standalone. So we will divide it to hardware and software parts, meaning the device and configuration software. The main features of the system that we are going to consider are:

- Capacity for 32 sensors inputs and 32 control outputs (32 alarm event channels).
- 12-key keyboard and 2-digit LED indicator is used to direct control interlock system.
- Two LED's for each alarm event channel. One is used to indicate alarm event while second shows the blocking state of this channel.
- Fast reaction to sensor signal changes. The response time must be less than 50ms.
- CPU supervisor circuit for resetting the system in case of internal failure.
- Remote configuration feature, including PC software. All alarm setpoints, comparison signs and handling algorithms can be adjusted.

Alarm system is constantly interacting with its environment. We can consider our device as a black box reacting to the requests and messages from the environment, which is composed of several agents. Each agent interacts with the system with a different purpose and it exchanges a different set of messages. We have identified three agents: the user, the sensors and devices under control (basically solenoid valves). The only one agent for configuration software is the user.

1.1 Use Cases

Use cases describe the functionality of the system from the agent's point of view. Each use case is a different way to use system and the completion of each use case produces a different result. Use cases for hardware and software parts are represented in the Figures 1 and 2.

Hardware use cases

Block alarm - the user enters an alarm event number using keyboard and indicator and presses the 'BLOCK' button. Selected alarm event won't be processed by the system.

Unblock alarm - the user enters an alarm event number using keyboard and indicator and presses the 'UNBLOCK' button. Blocking state of this channel is cleared. Selected alarm event will be processed by the system.

Block all alarms - the user enters a special code using keyboard and indicator and presses the 'BLOCK' button. This will set blocking state for all alarm channels. All alarm events won't be processed by the system.

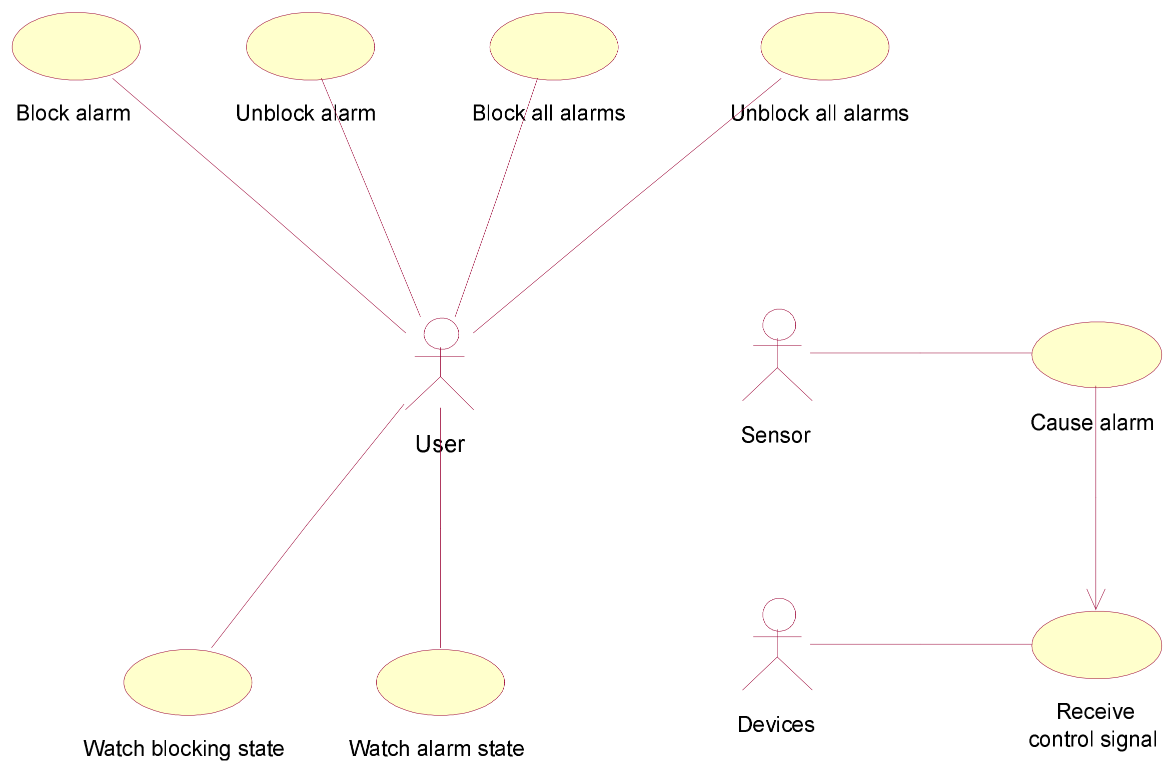

Fig. 1. Hardware use case diagram

Unblock all alarms - the user enters a special code using keyboard and indicator and presses the 'UNBLOCK' button. This will clear blocking state for all alarm channels. All alarm events will be processed by the system.

Watch blocking state - the system constantly shows blocking state for each alarm channel using the corresponding LED on front panel. The user just looks at it.

Watch alarm state - the system constantly shows alarm state for each channel using the corresponding LED on front panel.

Cause alarm - the sensor changes its' signal level exceeding the alarm setpoint and leads system to react.

Receive control signal - devices under control just get reasonable control signals and change their states.

Software use cases

Change averaging count - system is capable of averaging sensors analog signals in order to eliminate signal noise. Average count is configurable.

Change default valves configuration - the user can select default state for each device. This state will be used as normal, non-alarmed, state.

Read system information - the user can download status information from the device, like working time, number of supervisor resets, averaging count and default devices configuration.

Watch analog input values - the device keeps all sensor signal readings and they can be read by the user.

Change alarm setpoints and behavior - for each alarm channel there is a possibility to change setpoint, comparison sign and affected devices configuration.

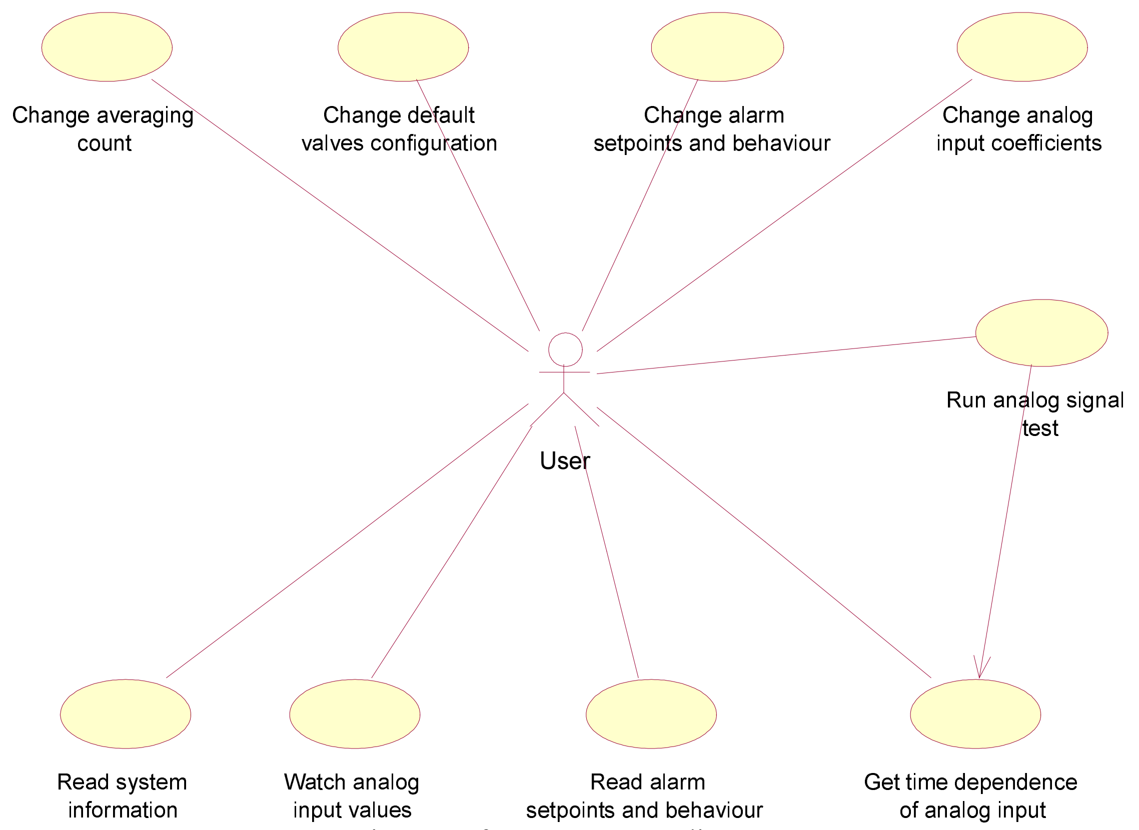
Change averaging count
Change default valves configuration
Change alarm setpoints and behaviour
Change analog input coefficients
Run analog signal test
User
Read system information
Watch analog input values
Read alarm setpoints and behaviour
Get time dependence of analog input

Fig. 2. Software use case diagram

Read alarm setpoints and behavior - the user can download alarms configuration from the device.

Change analog input coefficients - these are coefficients for conversion voltage to physical units, whereas the device operates only voltages but alarm setpoints are represented in physical units.

Run analog signal test - the user can enforce device to send time dependence of each analog channel.

Get time dependence of analog input - software shows received time dependence in a chart for each channel selected by user.

1.2 Scenarios

The scenarios should describe the interaction between the active external actors with the system. In our system the majority of scenarios are very simple and clear. There are only two most important diagrams: alarm handling diagram (Fig. 3) and alarm recovery diagram (Fig. 4).

The alarm handling specialty is as follows. The system stores affected devices pattern for all alarmed channel system. In the alarm recovery procedure the system changes only those outputs, which were not alarmed by the previous channels, making use of the stored list of the changed outputs. So the outputs which changed their states

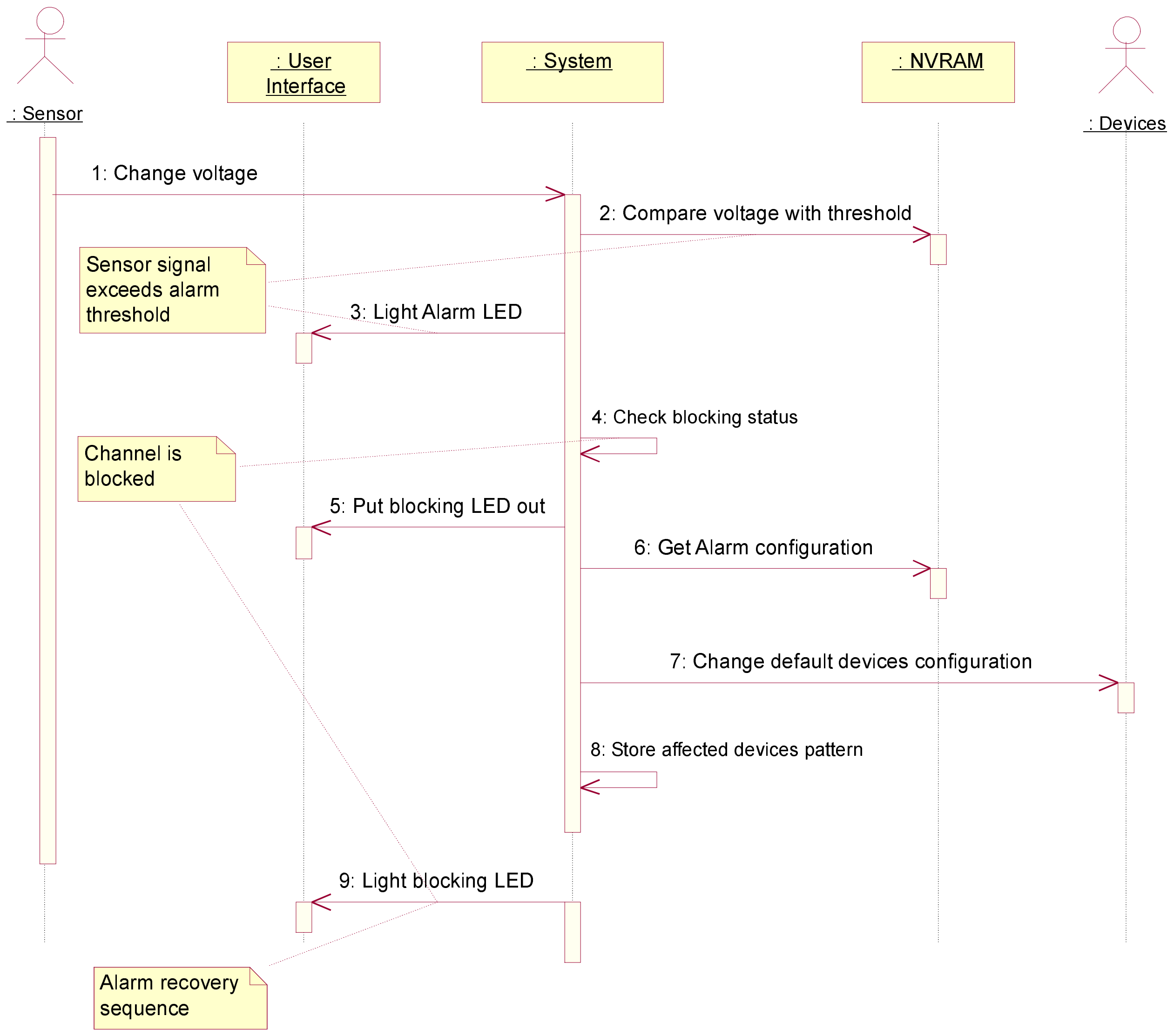

Fig. 3. Alarm handling sequence diagram.

because of the alarm are not affected by the alarm recovery procedure, if we consider a single processing (handling) cycle.

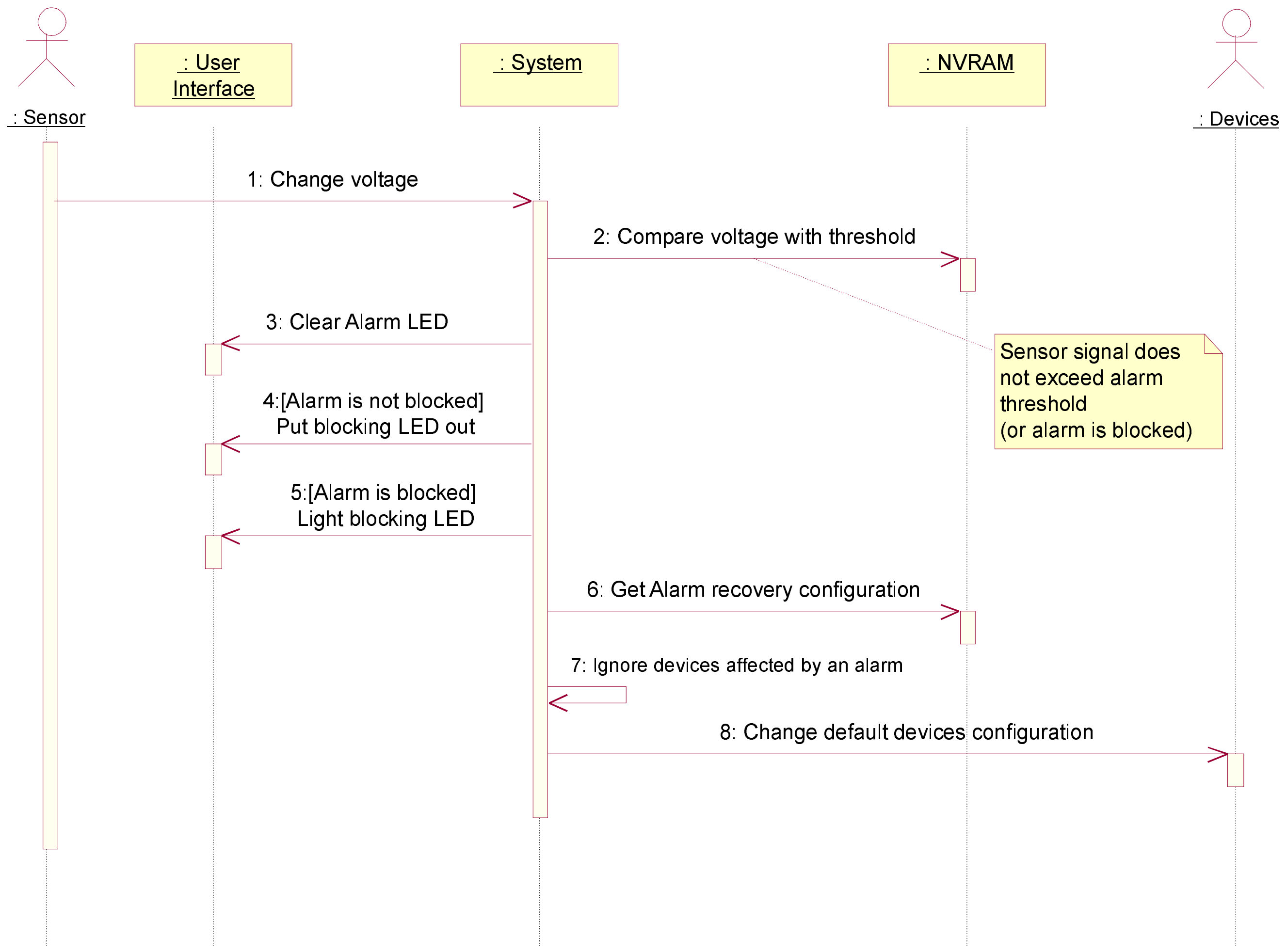
: Sensor
: User Interface
: System
: NVRAM
: Devices
1: Change voltage
2: Compare voltage with threshold
Sensor signal does not exceed alarm threshold (or alarm is blocked)
3: Clear Alarm LED
4:[Alarm is not blocked] Put blocking LED out
5:[Alarm is blocked] Light blocking LED
6: Get Alarm recovery configuration
7: Ignore devices affected by an alarm
8: Change default devices configuration

Fig. 4. Alarm recovery sequence diagram.

1. Object structure

Class diagram for PC software is shown in Fig. 5. The program contains 4 panes: System Information, Alarm Configuration, Analog Signal Test, Devices Configuration. For each pane it has corresponding object.

The alarms configuration data is kept in *Alarm Configuration* objects, while *S_Alarm_Configuration* class just gives a possibility to change them, download from alarm system or upload changed configuration. Besides, *S_Alarm_Configuration* is used for adjusting sensor parameters, like name, units and translate coefficients, which are helpful to represent all data in physical values, while alarm system operates only voltages. Sensor parameters are kept in the *SensorInfo* objects. Similar parameters describe each controlled device in the system (see the *DevicesInfo* class).

Software communicates with alarm system via RS-232 serial interface, using *S_Serial_Interface* class, which provides one command transfer with CRC check. The *S_NVRAM_Access* object makes it possible to transfer area of NVRAM specified by starting address and bytes number. This object transfer bytes with *S_Serial_Interface*.

User can read system information and analog inputs values using *S_System_Information* object. This object also operates averaging counter and does initial connection to alarm system, together with reading software version from it.

The *S_Analog_Signal_Test* intended to execute signal test procedure in alarm system, read signal histories and show them in chart. It also provides basic chart-handling procedures, like picture export.

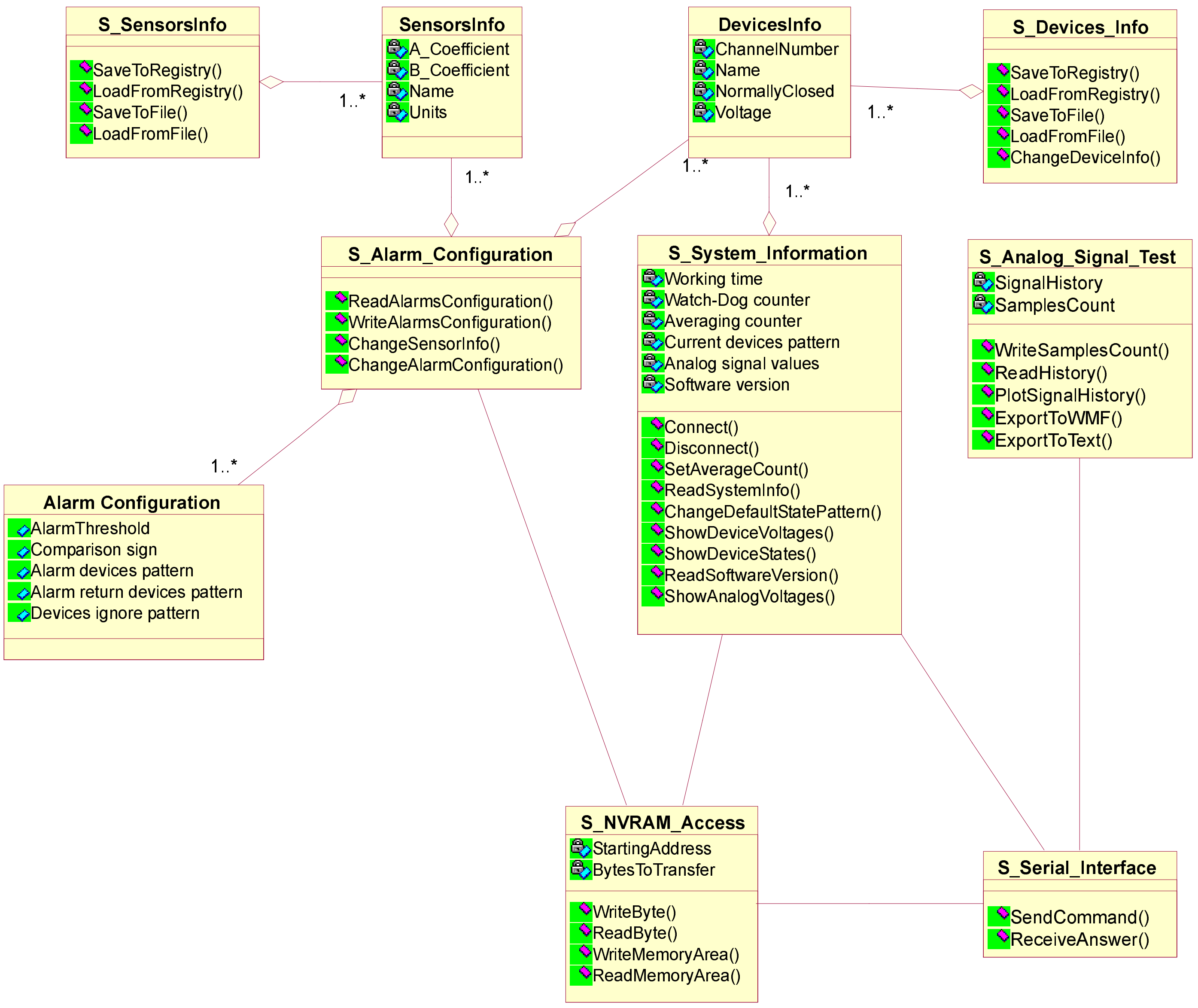

Fig. 5. PC Software class diagram.

The controller software class diagram is shown in Fig. 6. It also has the *Alarm Configuration* class, which is the same as one in PC software. All *Alarm Configuration* objects are kept in NVRAM object, which also contain all system parameters like *Working time, Averaging counter, Default state pattern etc*. The blocking statuses for each channel together with current analog signal values belong to NVRAM as well.

The software uses *Serial Interface* object that works as a communication server for PC software *S_Serial_Interface* object. All analog samples are read through *AnalogInput* class and devices are controlled with *DigitalOutput* class. The system also controls *User Interface*, which comprises *DigitalDisplay, Keyboard* and two LED displays (*Blocking LEDs* and *Alarm LEDs*).

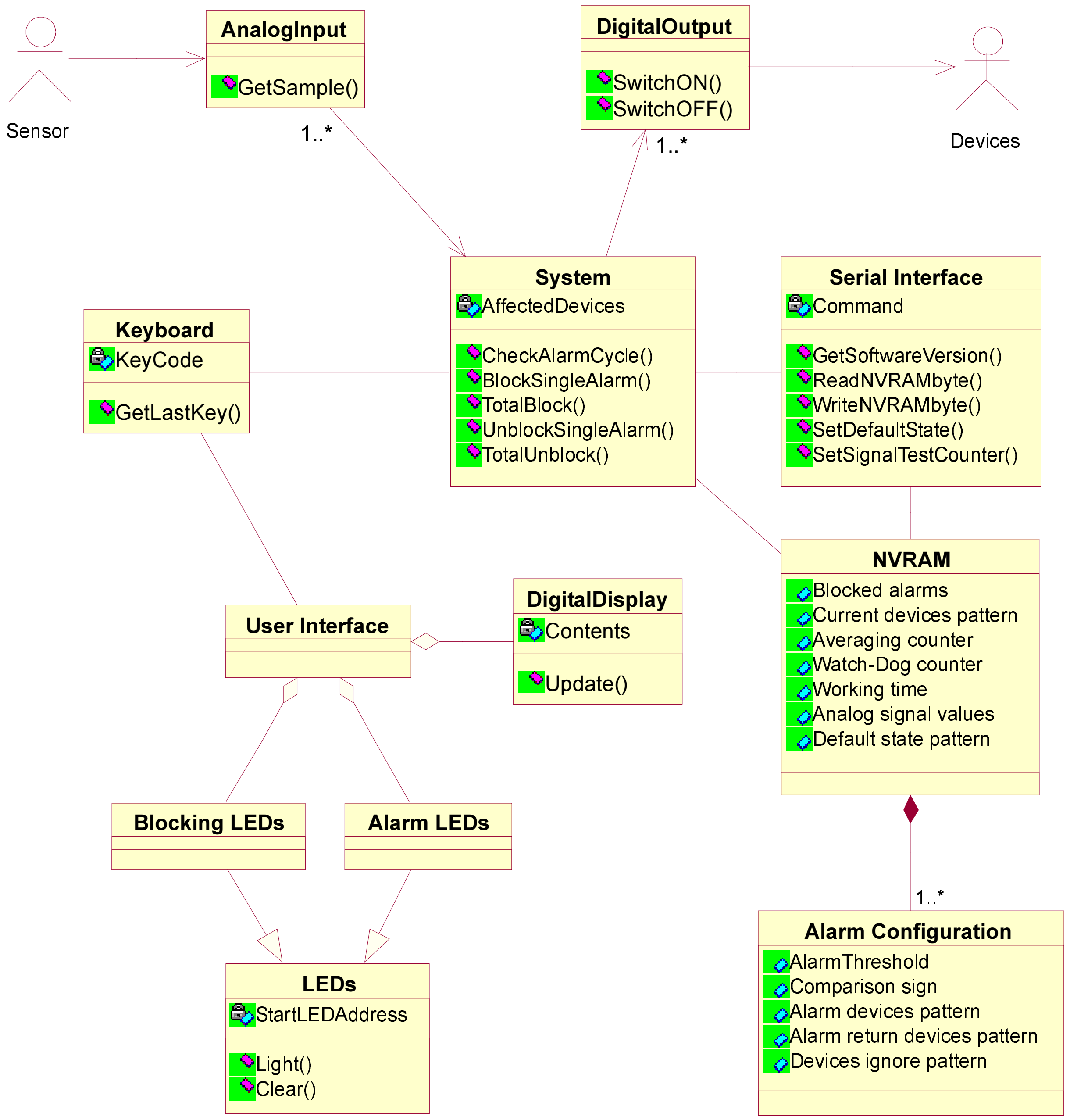

Fig. 6. Microcontroller software class diagram.