FeSCADA – Alarms

Motto: *If you think safety is expensive, try an accident!*
Stelios Haji-Ioannou

Introduction

The "Alarms" utility was added to FeSCADA software. With this feature the user can setup, monitor and record alarms events.

This chapter will cover the following topics:

1. Alarms basics
2. Alarm systems
3. Settings of alarms
4. Alarms and database
5. Application examples
6. Conclusions

1) Alarms basics

Alarms are audible and/or visible means of indicating to the operator an equipment malfunction, process deviation, or abnormal condition requiring a human response[1].

The necessity of implementing an alarm system derives from the complexity of the control systems used in automation today.

Poor alarm management systems expose companies, governments and the public to unnecessary hazards, environmental damage, loss of reputation, financial cost, injury and potentially loss of life[1].

The purpose of an alarm is to direct the operator's attention towards a plant condition requiring timely assessment and action (EEMUA, 2013). The alarm system should alert, inform, confirm and guide the operator with relevant alarms and useful information. Alarm annunciation should allow enough time for the operator to respond and prevent unnecessary shutdowns and limit process upsets (off specification product) which undermine profits. Alarms need to be purposeful and prioritized, highlighting critical conditions and provide detailed information for necessary response action to return the process variable back to the targeted set point value or operating envelope. It is important to note the operator is human; therefore consideration must include a rational expectation of one's ability to manage numerous abnormal situations in a short period of time[1].

Alarms are typically configured around an operational state i.e. when a pump is required to operate (run) any change in its run state, i.e. pumped overloaded and tripped, should trigger an alarm for the operator to investigate and correct the reason causing the pump to trip[1].

Naturally, some alarms are not necessary or are configured incorrectly, leading to nuisance alarms. Nuisance alarms do not assist the operator, they only crow the event log and load the controller's microprocessor[1].

2) Alarm systems

The term alarm system relates to the system as a whole including an alarm philosophy, alarm management, generation, logic, processing, alarm presentation and monitoring. A control system may be integrated with many dedicated control systems: safety instrumented system (SIS), emergency shutdown system (ESD), distributed control system (DCS) or basic process control system (BPCS) and Fire and Gas (F&G) system. Each controller communicates via dedicated networks to deliver alarms to the operator via HMI[1].

EEMUA publication 191 covers a large scope detailing the various aspects of alarm systems design, management and procurement. The amount of configurable alarms, their design and generation varies considerably[1].

Types of alarms (EEMUA publication 191)[1]:

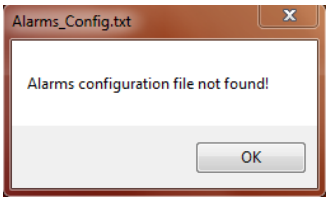
- **Absolute alarms:** compares the measured variable against a set point.
- **Bit-pattern alarms:** matches a predetermined pattern of digital signals.
- **Calculated alarms:** based on various signal and conditions (e.g. efficiency).
- **Control and instrumentation system alarms:** self-diagnosed faults within the control and instrumentation system hardware or software.
- **Deviation alarms:** compares signals for variation.
- **Discrepancy alarms:** generated by comparing an expected plant state against an actual plant state.
- **Rate-of-change alarms:** occurs when a rate of change exceeds a predetermined setting (e.g. heating speed).
- **Adaptive alarms:** generated using the 'rate-of-change' or 'deviation' principle in combination with absolute thresholds.
- **Adjustable alarms:** absolute alarms in which the alarm settings are adjusted to suit operating conditions.
- **Re-triggering alarms:** alarms which are automatically re-annunciate to the operator in certain conditions.
- **Recipe-driven alarms:** alarms that are turned on or off in different plant states.
- **Statistical alarms:** utilizing statistical calculations/process to filter out significant changes amongst noise.
- **First-up alarms:** used for examining the order of occurrence of alarms.

What should not be an alarm[1] ?

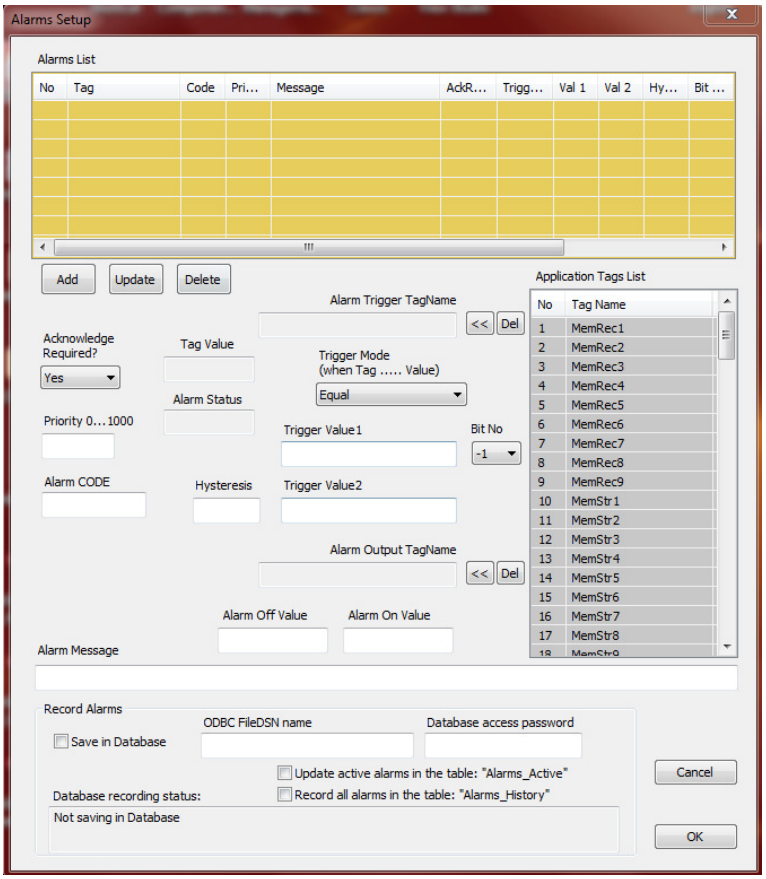
- Alarms without a defined operator response.
- Process variable or plant status changes that do not require the operator's attention.
- Events too fast for the operator to prevent.
- Events that are recorded in an alarm/event log which the operator does not need to see.
- Alarms to confirm successful operator actions.
- Duplicate alarms (may need to be logically suppressed).

3) Settings of Alarms

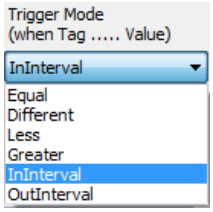
To open the **Alarms Setup** dialog window, select from the menu **Utils->Alarms Setup**. A notification message will inform if no data was previously configured. Press **OK** to open the dialog window. The **Alarms List** is on the top of the window. The **Application Tags List** is in the bottom right corner.



To setup an alarm first select the **Alarm Trigger TagName** from the **Application Tags List** using the **<<** button on the right of **Alarm Trigger TagName** text box. Once the tag is selected there can be two ways to check the value: bit level or whole value. To check at the bit level select a number from 0 to 31 from the combo box **Bit No**. In this case the **Trigger Value1** text box will take the value 1 and **Trigger Value2** will be empty. The **Trigger Mode** can have the options *Different* or *Equal*. If, for example, the alarm is to be active when the bit value is 1 (one) then *Equal* is the right choice. But if the alarm should be active when the bit value is 0 (zero) then *Different* is the right choice.



To check the whole value, select -1 for **Bit No**. In this case the **Trriger Value1** and **Trigger Value2** text boxes are enabled and will take any input from the user. That means the input can be a number or a string. The inputs are interpreted in relation with the tag type: integer, real or string. If the tag is integer or real, value1 and value2 are interpreted as numbers. If the tag is a string, value1 and value2 are interpreted as strings. **Trigger Value 1** is used in the comparisons: *Equal*, *Different*, *Less* and *Greater*. **Trigger Value2** doesn't need to exist for some comparisons. It is used only for the comparisons: *InInterval* and *OutInterval*. *InInterval* includes the values [value1,..., value2]. *OutInterval* does not include the values (-inf,..., value1) or (value2,..., +inf).



The alarm is becoming active when the comparison of the tag with value1 (and value2) is true.

Alarm **Acknowledge Required?** Can be *Yes* or *No*. If the option is *Yes*, the alarm will still be displayed even if the alarm is no more active (true). The operator has to see it. It will disappear only after the operator will acknowledge it, if the alarm is no more active. Historically, the alarms that require acknowledgment and that are not acknowledged would trigger an audible signal (siren, horn, ring bell, etc.), while the acknowledged alarms, or the alarms that do not require acknowledgment, would be just displayed on the HMI.

Priority and **Alarm CODE** are inputs with which the user can organize the alarms in groups and with different levels of emergency. These values are useful when analyzing the alarms history database.

Hysteresis is a threshold value that has to be surpassed for the alarm to deactivate. This helps to prevent chattering alarms. For example, if an alarm is activated when the temperature is greater than 100 degrees, and the hysteresis is 5, then the alarm will be deactivated when the temperature is less than 95 degrees. Observation!... If the temperature is less than 100 F (e.g. 97 F) and the operator is acknowledging the alarm, the alarm will deactivate, even if the hysteresis is not surpassed.

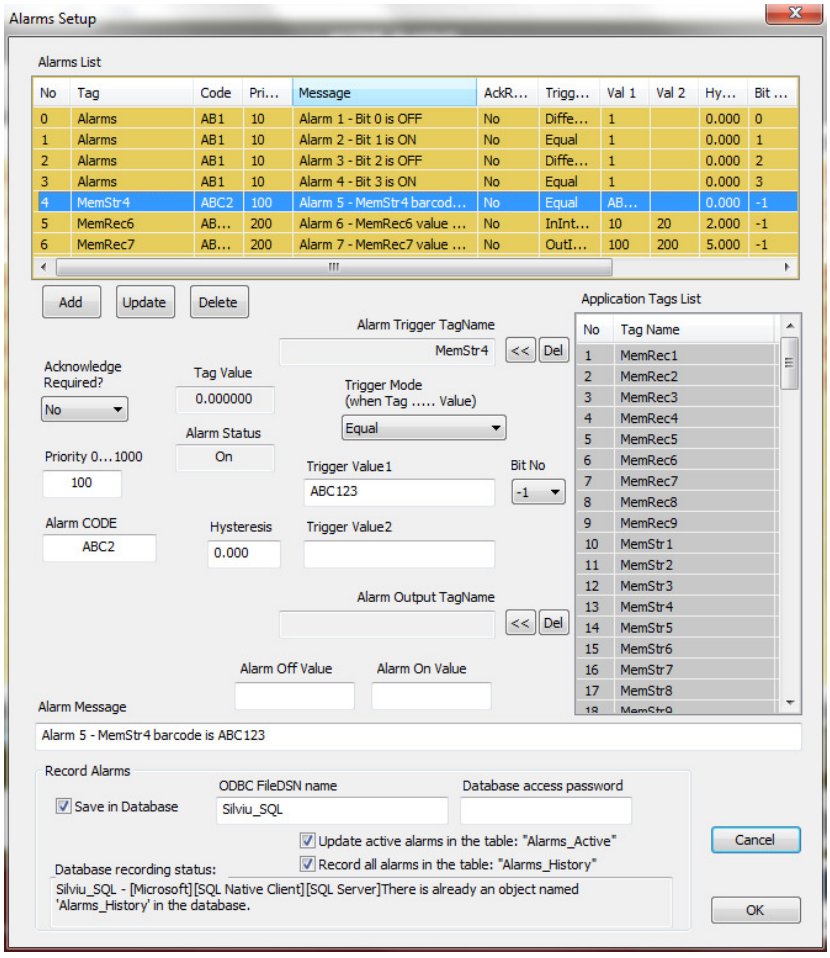
Alarm Output TagName is an optional tag that can have its value changed when the alarm becomes on, with the **Alarm On Value**, or when the alarm deactivates, with the **Alarm Off Value**.

Alarm Message is the text that will be displayed for the alarm. It is the explanation of the alarm, in words, for the operator / user.

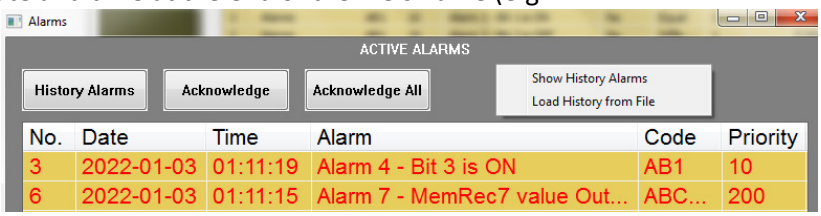
Tag Value and **Alarm Status** are read only text boxes to show the tag's value and the alarm status: on or off.

The user can add, update or delete an alarm with the buttons **Add**, **Update** and **Delete**. All data is saved in the configuration file when the user is pressing **OK** button to exit the setup dialog. Restart FeSCADA for the new alarms (or changes) to become active.

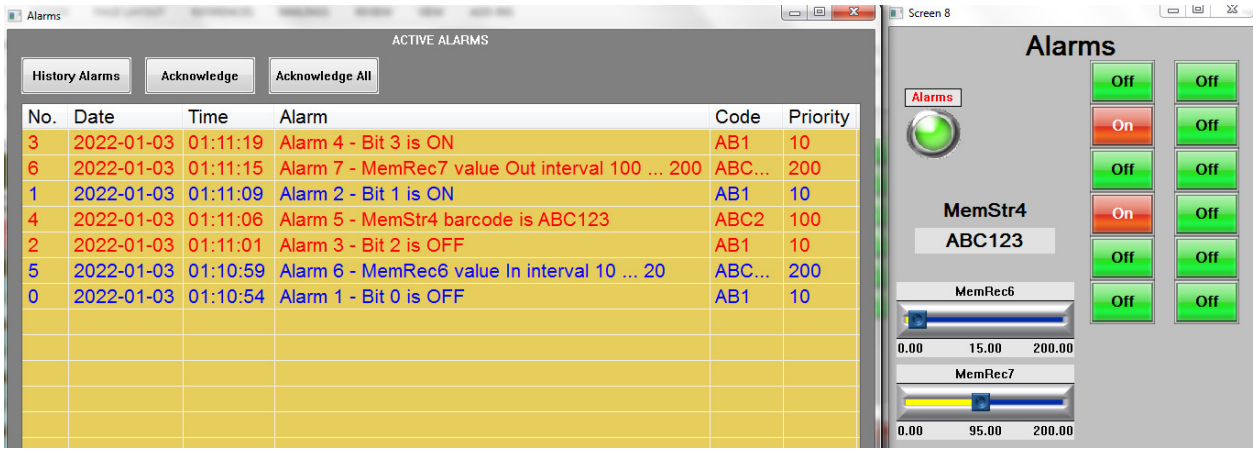
In the example below we setup 4 digital alarms (2 active when 0 and 2 active when 1), a string tag that will alarm when equal with "ABC123", and 2 real type tags to check the *InInterval* and *OutInterval* options. The *InInterval* has a hysteresis of 2 and the *OutInterval* has a hysteresis of 5.



By default, the alarms are saved in the text file *"...\ALARMS\Alarms.log"*. When the file becomes bigger than 1MB the file is saved with the current date and time at the end of the file's name (e.g. Alarms_2021_10_25_19_55_30.log) and another *Alarms.log* file is created.

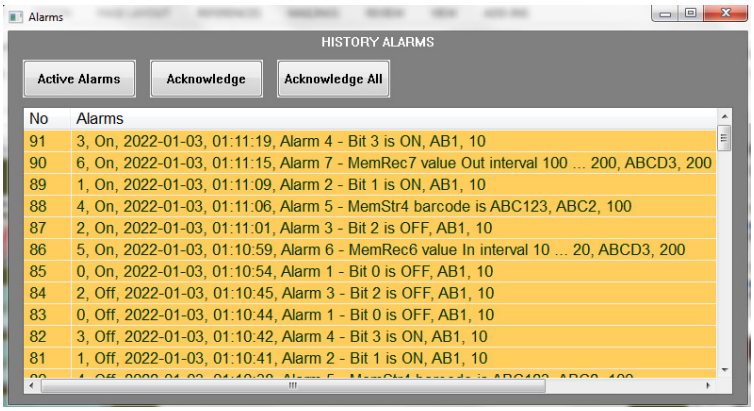


To see the active alarms the user can open the window **Alarms**, from the main menu. The see the alarms history press the button **History Alarms**. The alarms from the file *Alarms.log* are displayed in the list view table. To see older alarms right click inside the window and select **Load History from File**.



When an alarm is acknowledged but still active the color is changing from red to blue. The user has the option to acknowledge one alarm at a time or to acknowledge all alarms that are active at that moment.

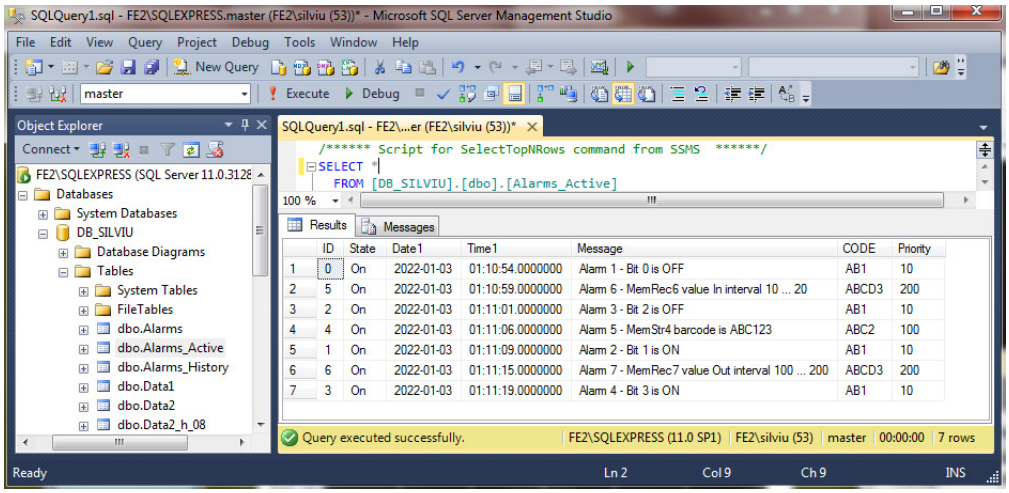
If the user is defining a memory tag with the name *SysAlarms*, the total number of active alarms will be written in this tag.

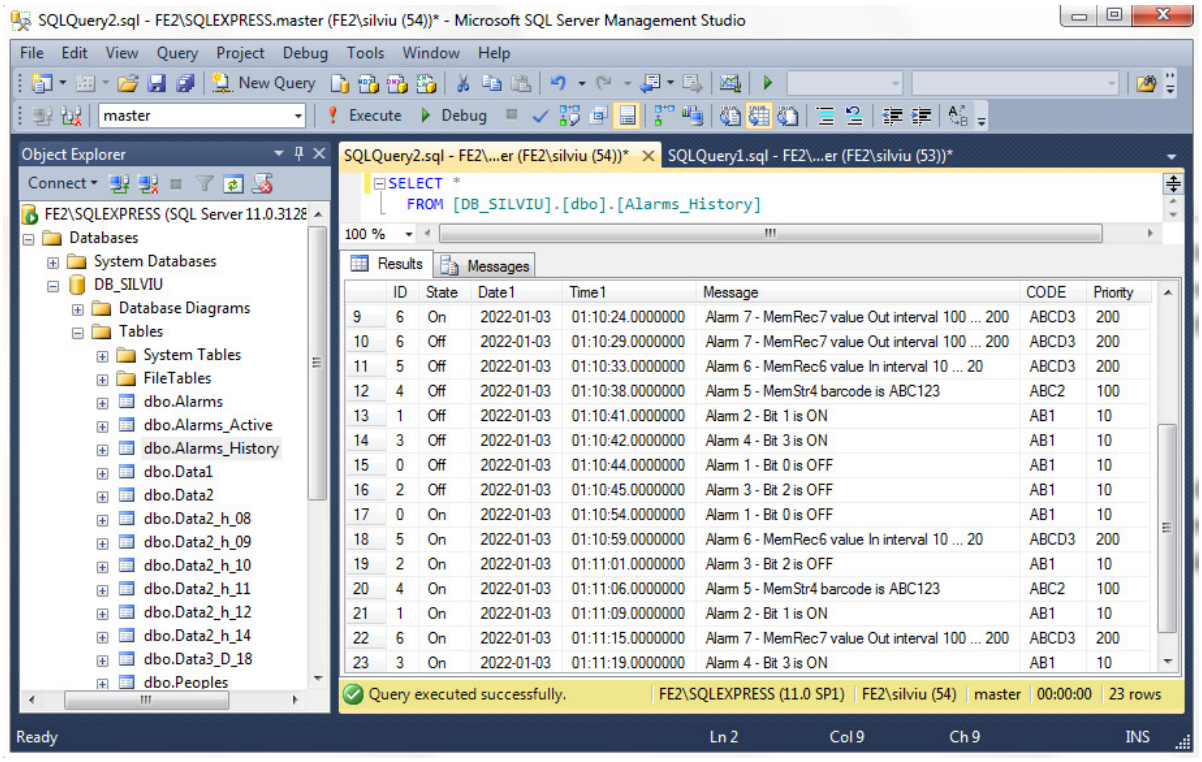


4) Alarms and databases

On the local computer FeSCADA will always save the alarm events in the file *"..\ALARMS\Alarms.log"*. But there are cases when the alarms have to be recorded in a database which can be on another computer, in the network. FeSCADA has the option to record in a database defined in a File DNS, ODBC. It can update only the active alarms in the table *Alarms_Active*, it can record all the alarm events (On, Off) in the table *Alarms_History*, or can do both.

In the pictures below there is an example of using a Microsoft SQL database to update the active alarms and to record the alarms history from the example above.





5) Application examples

There are not too many process/machine monitoring applications without alarms.

The first level of alarms is the connection and communication with different remote terminal units and PLCs. The second is the health of the instrumentation (sensors) and actuators used in the automation. The third level is the process/machine itself. The second and third level are implemented usually at the PLC level and are sent to the SCADA software as an array of bits or values.

The fourth level are advanced alarms, used to predict if the production output is as expected for the current shift, or to perform preventive maintenance, or to make pre-programed revisions for parts of the machine, etc.

6) Conclusions

With FeSCADA software it is possible to setup and run alarms. An alarm is linked with a tag's discrete bit number or value. Bit level check or mathematical comparisons are used to evaluate if an alarm is active (On) or not active (Off). A code, a priority and a text message are associated with any alarm. Alarms are recorded on the local computer and optionally in a database, defined by the user.

References:

1. Ross Guy, 2016, ***"Best practice management of industrial process control alarm floods"***, ENG4111 and 4112 Research Project, University of Southern Queensland, Faculty of Health, Engineering and Sciences.