

## Table of Contents

<b>6</b>	<b>Alarms .....</b>	<b>6-3</b>
6.1	All About Alarms .....	6-3
	Where the Alarm Processing is Carried Out .....	6-4
	Information You Need to Configure Alarms .....	6-5
	Intelligent Alarm Features.....	6-7
	How the Alarm Manager Works .....	6-9
	Starting the Alarm Manager.....	6-10
	Restarting the Alarm Manager .....	6-11
	Alarm Specification.....	6-11
	How the messages are stored.....	6-12
	Alarm Summary Displays.....	6-14
	Alarm Printout .....	6-15
	How Alarm Buzzers are linked to the Keyboard .....	6-16
	How the Alarm Buzzers Work .....	6-16
6.2	The Alarms Configuration Screen .....	6-17
6.3	Alarm Configuration.....	6-18
	Alarm Configuration (Entity & Attribute) .....	6-19
	Alarm Configuration (Type) .....	6-20
	Alarm Configuration (Expression) .....	6-21
	Alarm Type Configuration.....	6-22
	Report by Exception Alarms .....	6-24
	Alarm Configuration (Deadband).....	6-25
	Alarm Configuration (Priority).....	6-26
	Alarm Configuration (Enabled, Group and Delay).....	6-27
	Alarm Configuration (Display) .....	6-28
	Alarm Configuration (Message and Colour) .....	6-29
	Alarm Configuration (Action) (UNIX System) .....	6-30
	Alarm Configuration (Action) (NT System) .....	6-31
6.4	Arguments passed to the action program (UNIX System) .....	6-32
	Action Example (UNIX System).....	6-33

	Generic Example (UNIX System) .....	6-33
6.5	Arguments passed to the action program (NT System).....	6-35
	Action Example (NT System) .....	6-36
	Generic Example (NT System).....	6-36
6.6	Checking out the Alarm Configuration .....	6-38
6.7	Navigator Alarms Colours .....	6-39

### **List of Tables**

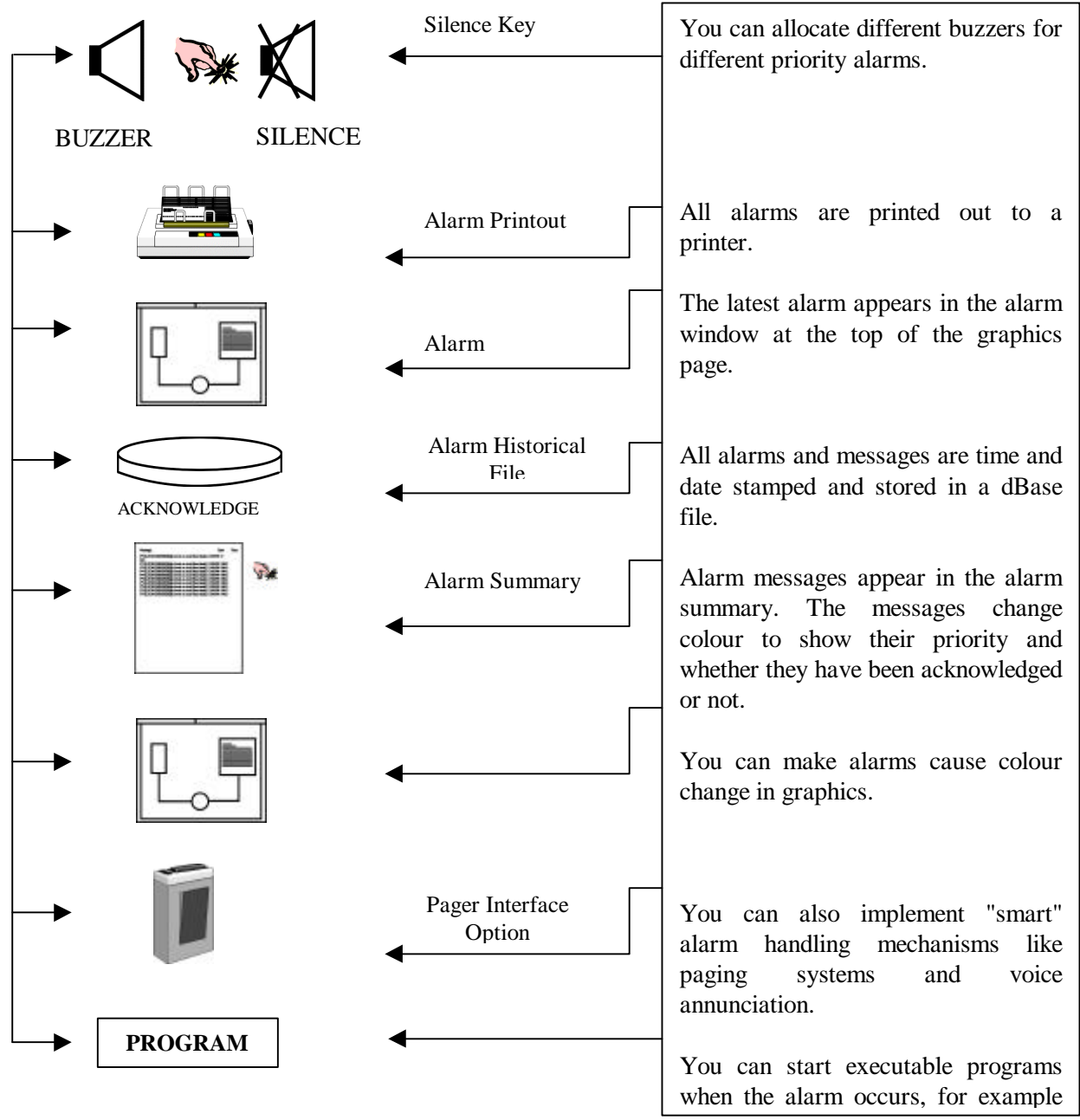
<i>Table 1: Alarm Specification Fields</i> .....	6-5
<i>Table 2 : Location of alarm storage</i> .....	6-12
<i>Table 3: Alarm Message Database Structure</i> .....	6-12
<i>Table 4: Alarm Type Configuration</i> .....	6-22
<i>Table 5: Arguments to be passed to a UNIX script</i> .....	6-32
<i>Table 6: Arguments to be passed to a NT script</i> .....	6-35
<i>Table 7: Default Alarm Colours in the Navigator</i> .....	6-39

# 6 Alarms

## 6.1 All About Alarms

As part of the *MacroView* package, there is a complete alarm system that continuously scans the data for alarm conditions. The diagram shows the events that take place when an alarm is detected.

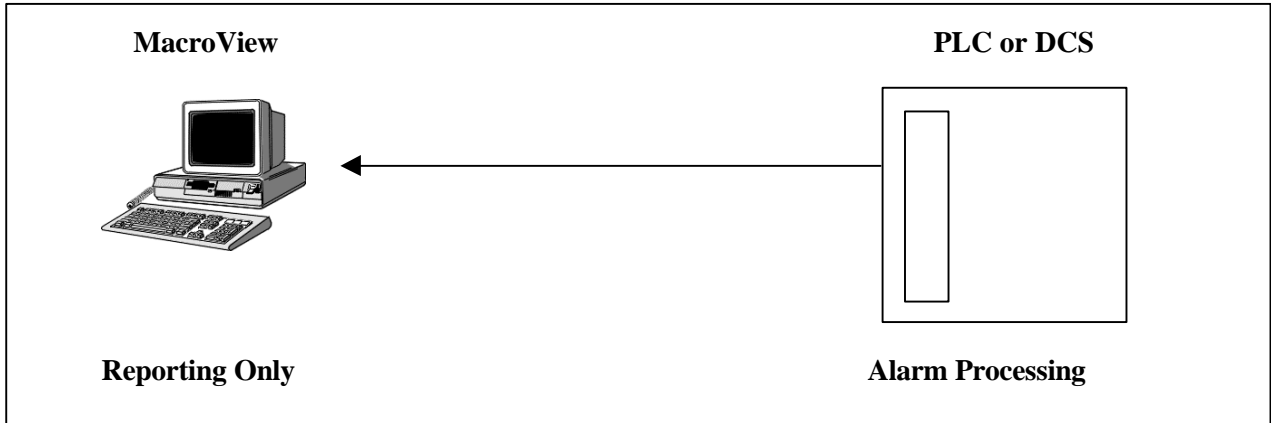
An alarm is produced by a scanned value going into alarm or report by exception alarm or a program.  
**Note:** The data used for the alarm can originate from any source, e.g. PLC, DCS, TEXTFILE, DBASE Source etc.



## Where the Alarm Processing is Carried Out

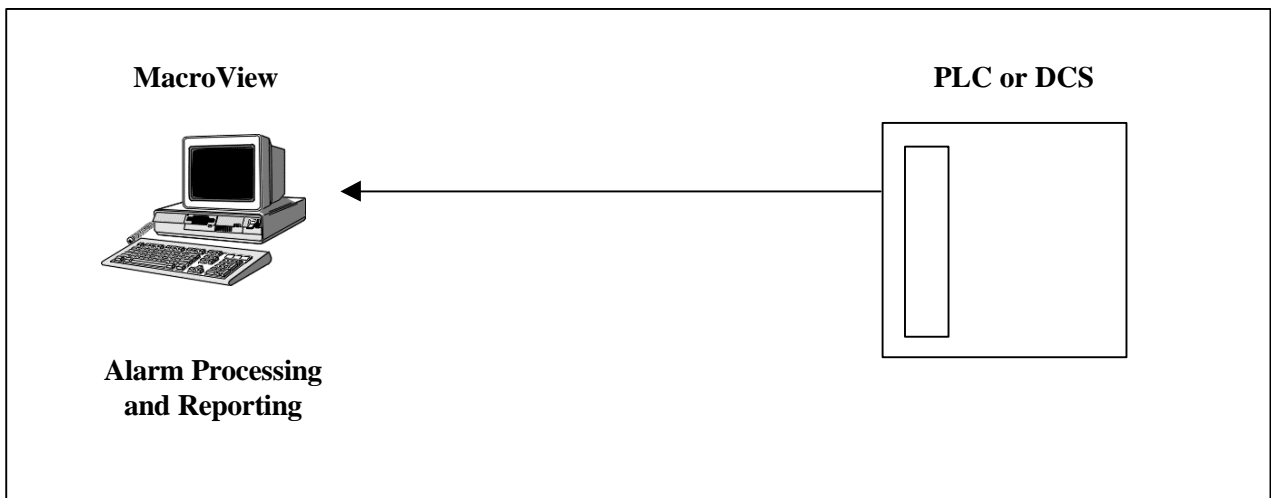
Before you configure the alarm system, take a moment to consider the best alarm strategy. Most important, decide where you want to do the alarm processing. *MacroView* supports three options:

**i. Alarm processing in the DCS or PLC**



In this case, *MacroView* simply reports on the alarms that are detected and signalled to *MacroView*. This is often the preferred option if the front-end equipment has the capability because it is a report-by-exception philosophy. (See the section "Report By Exception Alarms" on page22 in this Chapter.)

**ii. Alarm Processing in *MacroView***



In this case, *MacroView* scans the live data and carries out all the processing and reporting. This is a simple and flexible approach. Clearly the speed at which the *MacroView* system reacts to alarms is dependent on the speed of the *MacroView* computer and on the speed of the communication link. **Note:** Be aware that your *MacroView* processor has more of a load on it using the scanning approach.

iii. **Advanced Processing in *MacroView*, Basic Processing in PLC/DCS**



This technique results in the basic processing of alarms being carried out quickly and efficiently in the PLC or DCS and the more intelligent alarms being carried out in

***MacroView***

Often, where you have multiple sources, you will be able to do report-by-exception on some equipment and you have to do alarm scanning on other equipment.

**Information You Need to Configure Alarms**

*MacroView* gives you the ability to freely characterise the alarm system so that it closely matches your requirement.

To do this, you must configure various fields in a database for each alarm in the system.

The alarm manager then scans this database and reacts according to your entries in the fields.

The characteristic of each alarm is stored in a record in a dBase file called `almspec.dbf` in your configuration directory.

The names of the fields and their meaning are as follows:

**Table 1: Alarm Specification Fields**

Field Name	Description	dBase Format
ENTITY	The entity to which the alarm is associated.	Character 8
ATTR	The attribute that is to be used in the alarm decision.	Character 8
ALMTYPE	The alarm type whether HIGH, HIHI, DEVN, REPORT etc. The Alarm type is a critical field. What you choose here will affect the format of some of the other fields.	Character 8
EXPRN	The threshold value at which the alarm becomes active. (May also contain a free-format expression.)	Character 48
PRIORITY	The alarm priority (1 to 5). Different buzzers and actions occur for the different alarm priorities.	Numeric 2

Field Name	Description	dBase Format
DEADBAND	A setting in percent (typically 1 or 2%) that stops repetitive cycling of an alarm that is just on the threshold	Numeric 3.1
ENABLED	Whether the alarm is enabled or not.	Logical 1
GROUP	The alarm group in which the alarm belongs; groups of alarms may be collectively enabled and disabled	Numeric 3
DISPLAY	The schematic most useful in diagnosing the cause of the alarm. This graphic can be called up directly when the alarm first occurs.	Character 10
MESSAGE	A 32 character free-format message that appears in the alarm historical log and in the alarm summary display.	Character 32
COLOUR	The colour of the alarm message - may be coded for different priorities or alarm types.	Numeric 2
ACTION	A supervisory program to be executed when the alarm occurs.	Character 32
DELAY	The number of seconds after the alarm condition occurs before the alarm manager recognises and reports the alarm. This is used in eliminating short fleeting nuisance alarms.	Numeric 3

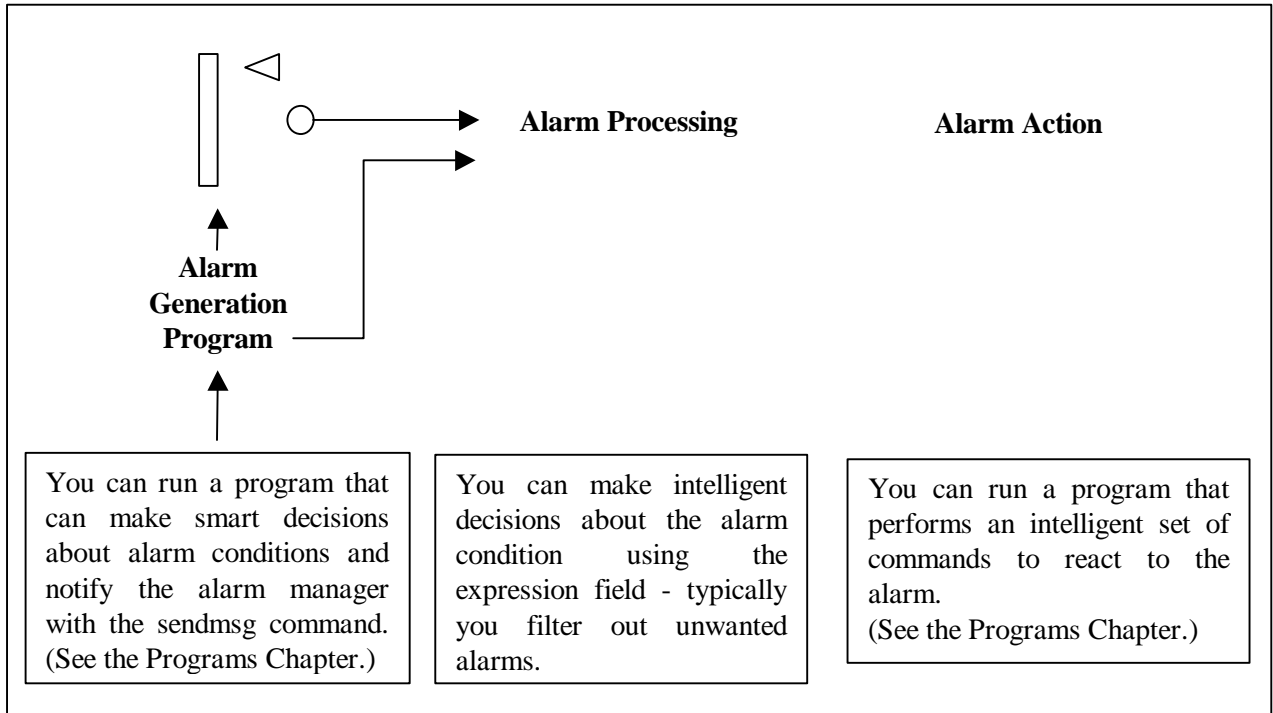
All of these fields are discussed in more detail in the configuration section later in this chapter.

**Note:** Unlike most process systems which offer a certain number of predefined alarms per entity, *MacroView* has no software imposed limit on the number of alarms or alarm Types per entity. In database terms, *MacroView* assigns a separate record to each alarm as opposed to most other systems that reserve fields in their tag database for each alarm.

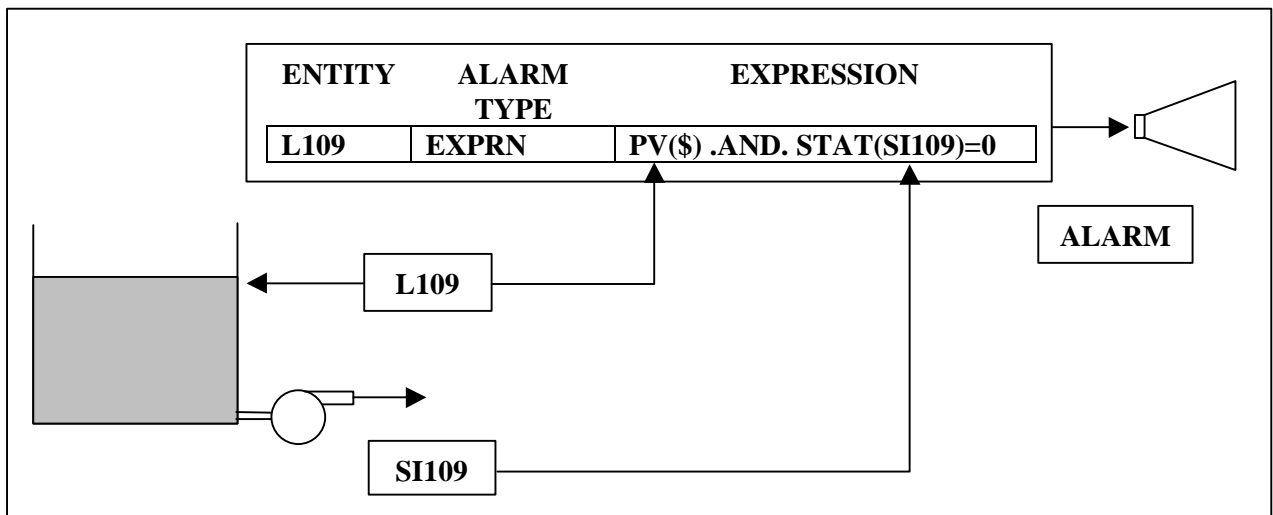
## Intelligent Alarm Features

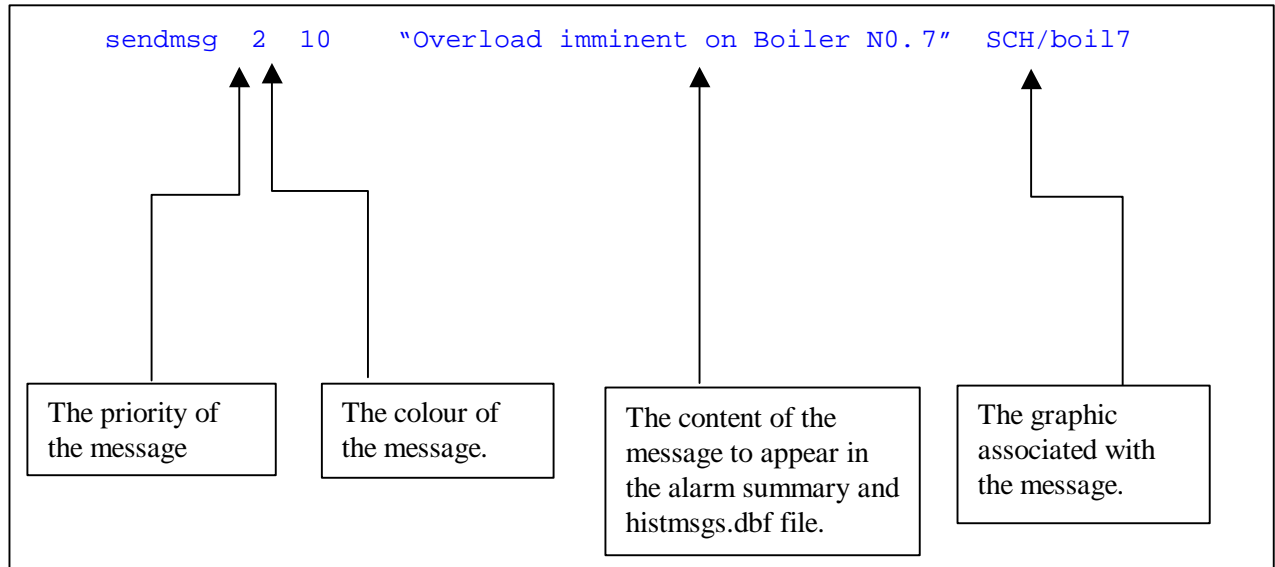
One of the major drawbacks of most alarm systems is that they produce too many alarms. This has the effect that the operators tend to ignore the alarms and usually miss the critical ones. *MacroView* provides the facility to add intelligence to the alarm system so that the "false or trivial" alarms such as those created by off-line equipment can be filtered out.

You can add intelligence to the alarm system in one of three places.



The diagram below shows how the expression field can be used in a typical alarm situation. In this example we only want an alarm if the level is high and the pump has not turned on.



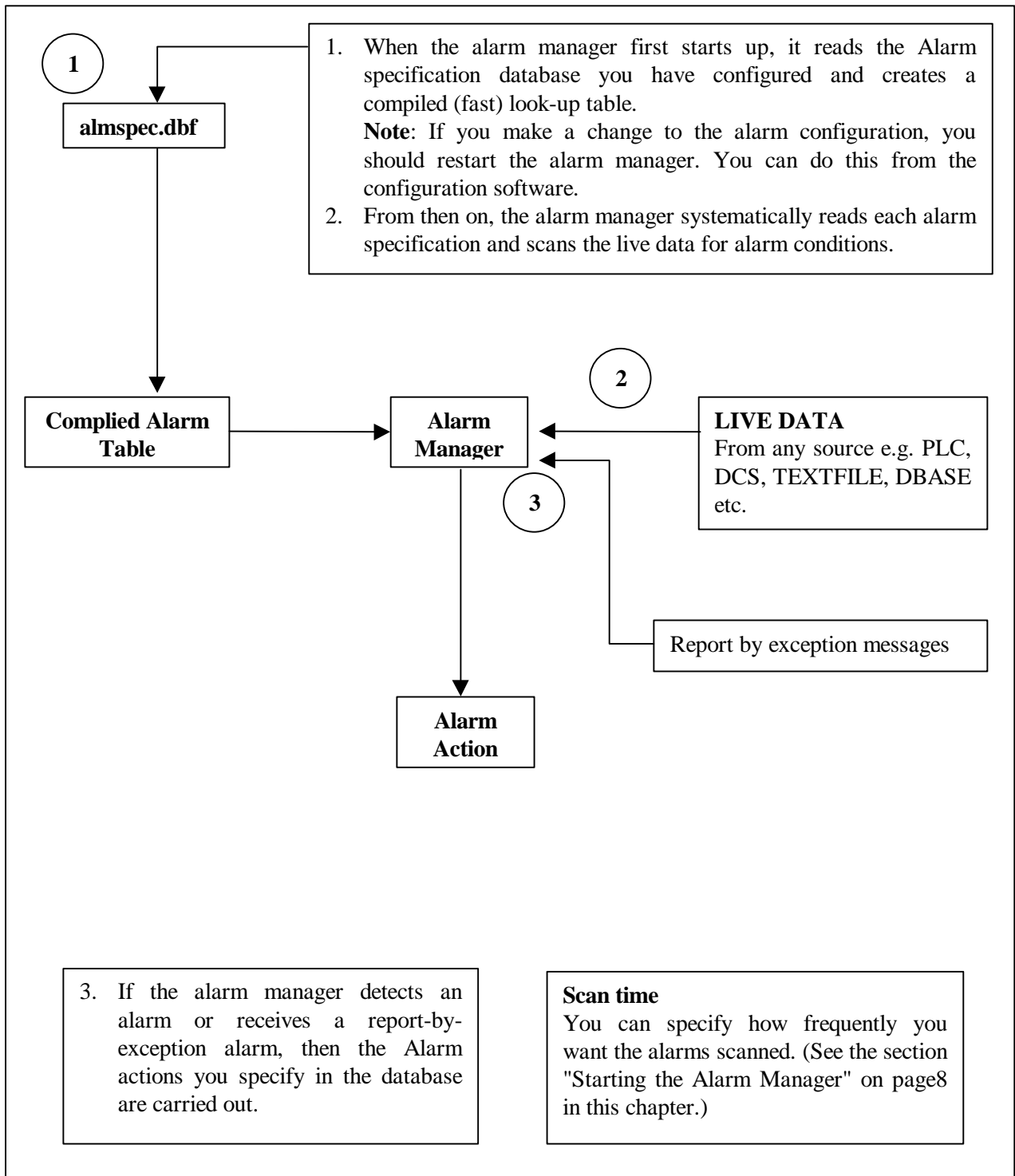


Typical messages from a high level program running a process model would look like this:

This type of message can be sent from a dBase program, a shell script program etc.

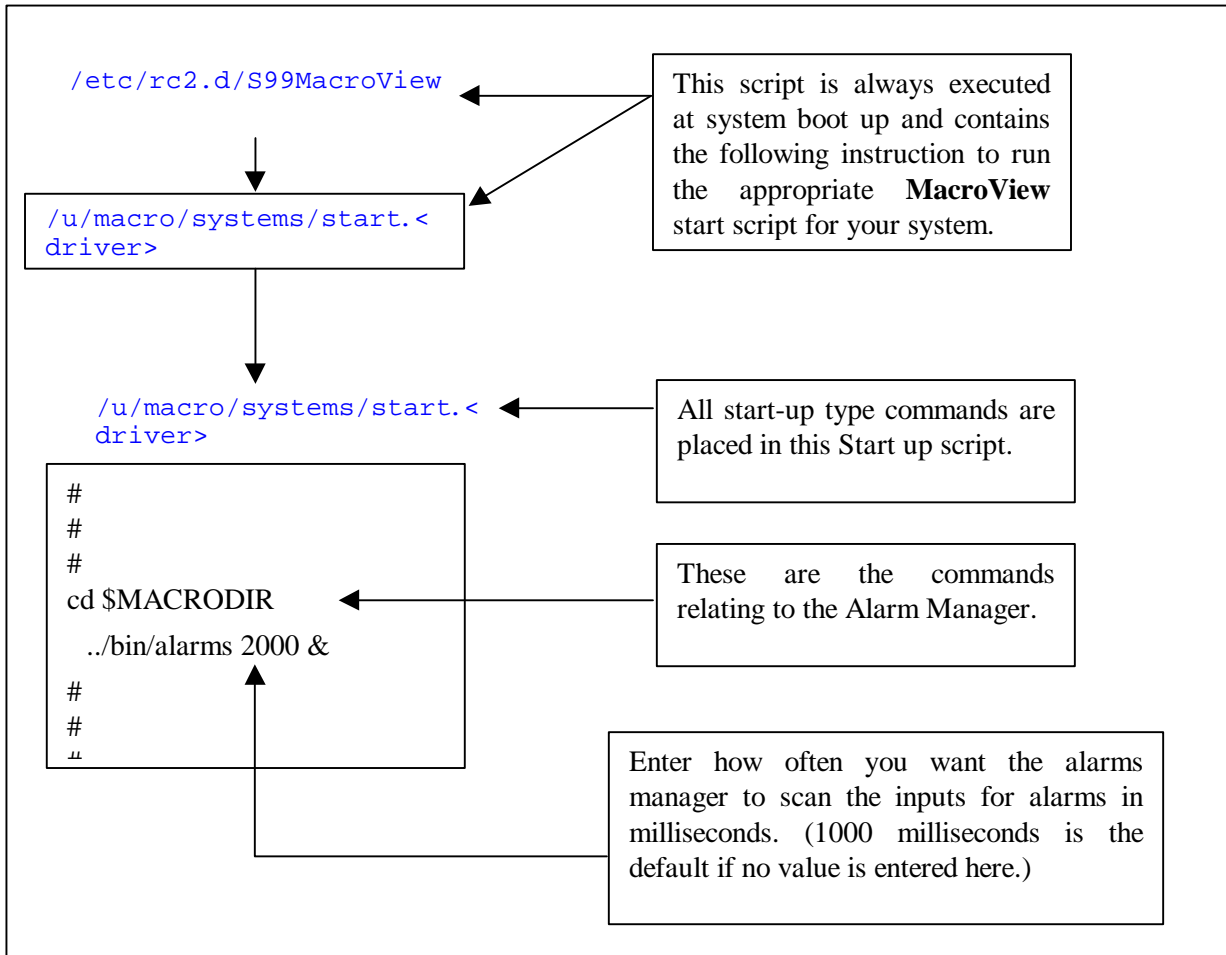
## How the Alarm Manager Works

The diagram below shows in concept how the alarm manager works.



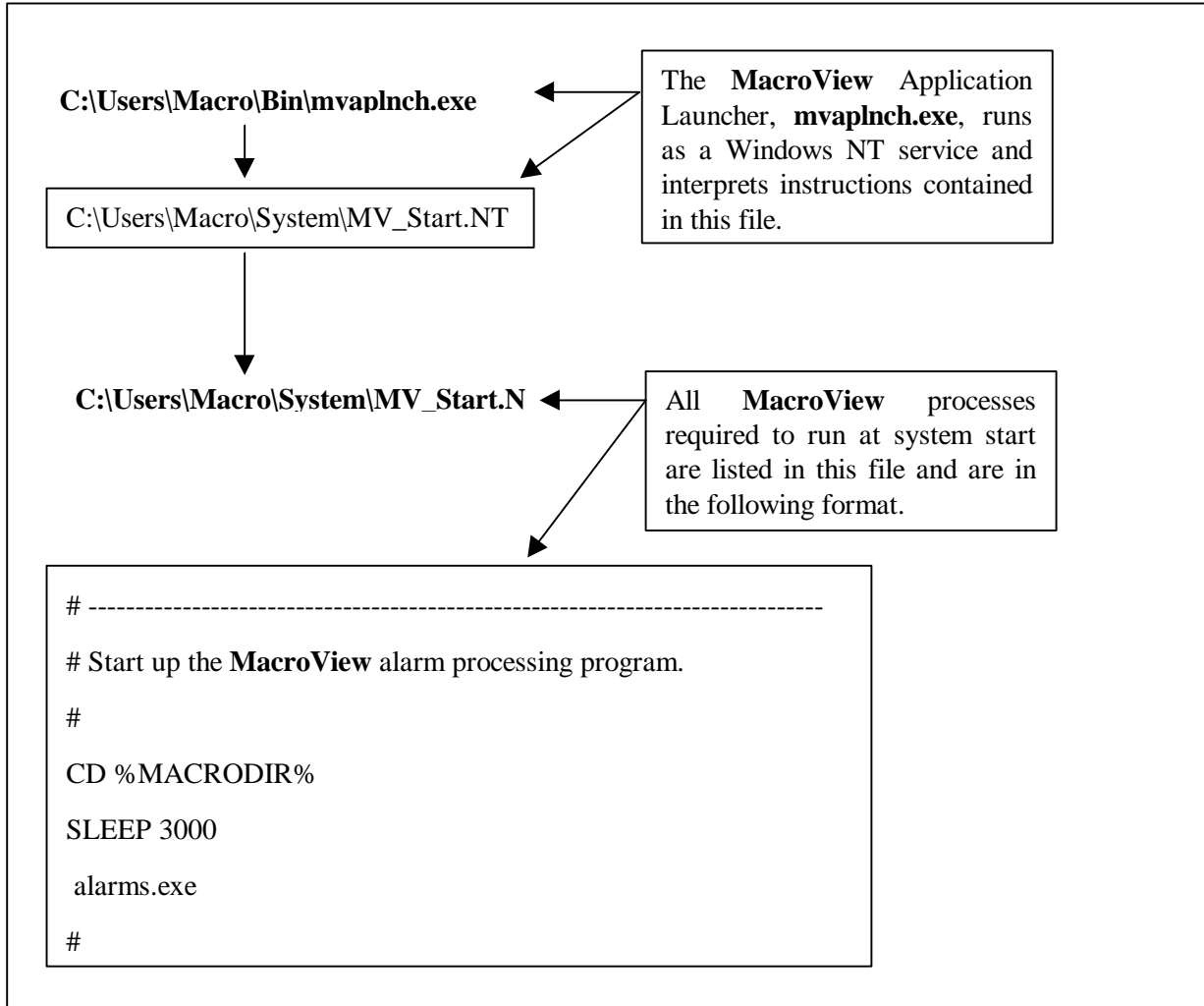
## Starting the Alarm Manager

The Alarm Manager program is usually started when the system is first switched on and remains running at all times. The system administrator is usually responsible for ensuring this happens. The method for achieving this differs slightly between a UNIX and NT system. In a UNIX system, the History Manager is started with a scenario like the one shown in the diagram below. (The example is for a UNIX system, such as SCO UNIX).



In an NT system, background processes (known as daemons in a UNIX system) are run as Windows NT services. The **MacroView** Application Launcher, [mvaplunch.exe](#), runs as a Windows NT service and is controlled through the Service Control Manager (SCM).

The file `%HOME%\System\MV_Start.NT` provides the processes to be started, and is interpreted by the *MacroView* Application Launcher. This file contains the instructions to start such processes as the History Manager, Alarms Manager and drivers etc. The diagram below explains the concept and more detailed information can be found in the Server Installation Manual for Windows NT, (document number **IM-NTS-310**).



## Restarting the Alarm Manager

For reasons of processor efficiency, the Alarm Manager does not continuously read the `almspec.dbf`, instead, it reads this file when it first starts up and compiles an image of it, which it uses from then on. This means that changes you make to this file are not recognised until you restart the alarm manager. We therefore recommend the following procedure.

## Alarm Specification

If you make any changes to the Alarm specification data base (`almspec.dbf`), you should select the *Alarming:Restart Alarm Manager* option from the Engineering Configurator, or alternatively, you can just type `restart alarm` at a UNIX command line prompt or DOS command line prompt, depending on your system type. This command does not actually stop and restart the Alarm Manager program, however, it does send a signal to it, which causes it to read the alarm specification database again and create a new image of it in memory.

## How the messages are stored

When the alarm manager detects an alarm, it stores the information in dBase files in the `msgs` directory.

The `msgs` directory is a subdirectory of your home directory. The table below shows where the alarm manager stores the information

**Note:** Each of these databases has a corresponding `.mdx` index.

**Table 2 : Location of alarm storage**

Filename	Types of Alarms stored	Priority
histmsgs.dbf	All alarms including log only messages	All priorities
sysalarm.dbf	System Alarms	0
alarms.dbf	Emergency alarms	1
	Alarms	2
	Warnings	3
guides.dbf	Operator Guide Messages	4

The structures of each of the databases are the same for all these databases and are as follows:

**Table 3: Alarm Message Database Structure**

Structure Field	Field Name	Type	Width	Description
1	DT	Date	8	The date the alarm occurred.
2	TM	Character	8	The time the alarm occurred.
3	SRCDT	Date	8	The time the alarm occurred at measured by the source.
4	SRCTM	Character	8	The time the alarm occurred at measured by the source.
5	PRIORITY	Character	2	The priority.
6	COLOR	Numeric	3	The colour of the message.
7	AREANAME	Character	8	The area associated with the entity in the alarm.
8	ENTNAME	Character	8	The name of the entity in the alarm state.
9	ATTRNAME	Character	12	The attribute associated with the alarm.

Table 3: Alarm Message Database Structure

Structure Field	Field Name	Type	Width	Description
10	EVENT	Character	10	INTO or OUT OF
11	STATE	Character	10	HI, HI HI, Low etc.
12	ACK	Logical	1	Whether the alarm has been acknowledged or not.
13	ALARM	Logical	1	Whether the entity.attr is currently in alarm or not.
14	SUPERFIND	Character	12	The display or area associated with the alarm.
15	MSG	Character	64	The message including entity.attr that was sent to the alarm manager.

- The [alarms.dbf](#) contains only those alarms and messages that have
  - i. **Not been acknowledged, or**
  - ii. **Have been acknowledged but are still in the Alarm State.**
- [sysalarm](#) and guides records are only removed manually. I.e., They don't have "OUT OF" alarm states.

I.e. The alarms are only removed when they have been acknowledged and they are no longer in the Alarm State.

- The [histmsgs.dbf](#) file contains a record of all types of alarms, operator guide messages and return to normal messages. The file size is rotary, i.e. it builds up to a maximum size and then rewrites over the oldest alarms. The size of this file is 1000 records by default but it can be set to any size by changing a command line argument for the [mvmsgs](#) program. See the [mvmsgs \(C\)](#) document. This will need to be updated if it exists.

## Alarm Summary Displays

The Alarm summary displays the alarms in a scrollable browse widget. There are various controls that you can use to display the alarms in different ways. Each alarm is date and time stamped and colour coded. From the navigator press A to bring up the alarm summary.

The diagram below shows the alarms ordered by priority with acknowledged alarms hidden.

### Alarm Summary

Date/Time Stamp

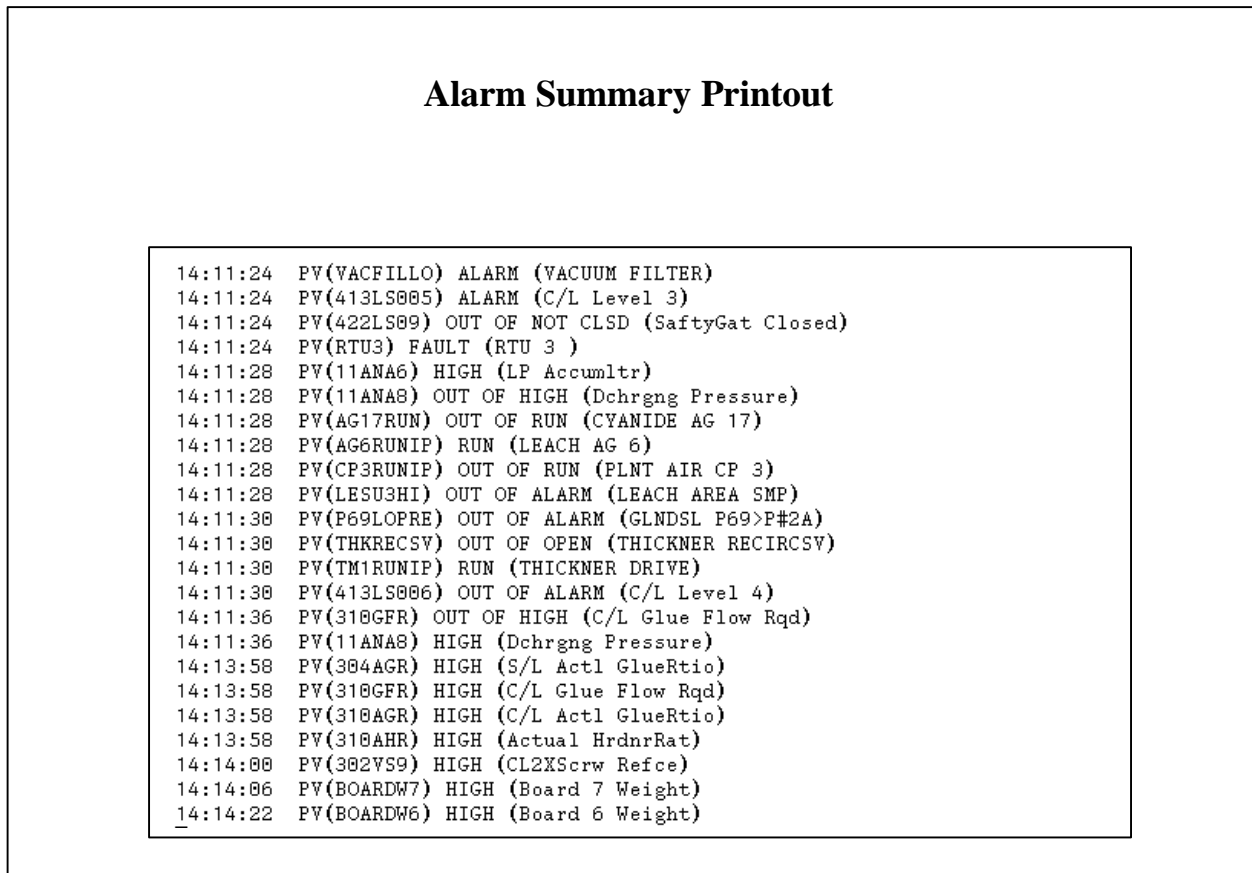
Entity Name

Free format alarm description, the colour may be configured to your requirements.

**Note:** You can configure the Return to Normal messages to occur in the alarm summaries. They also appear in the alarm window and in the `histmsg.dbf` file. See `mvmsgs(C)`  
 An alarm will be automatically be removed from the alarm summary when it has been acknowledged and it has returned to normal.  
 There are separate pages (and keys) for the system alarms, Messages and Alarms.

## Alarm Printout

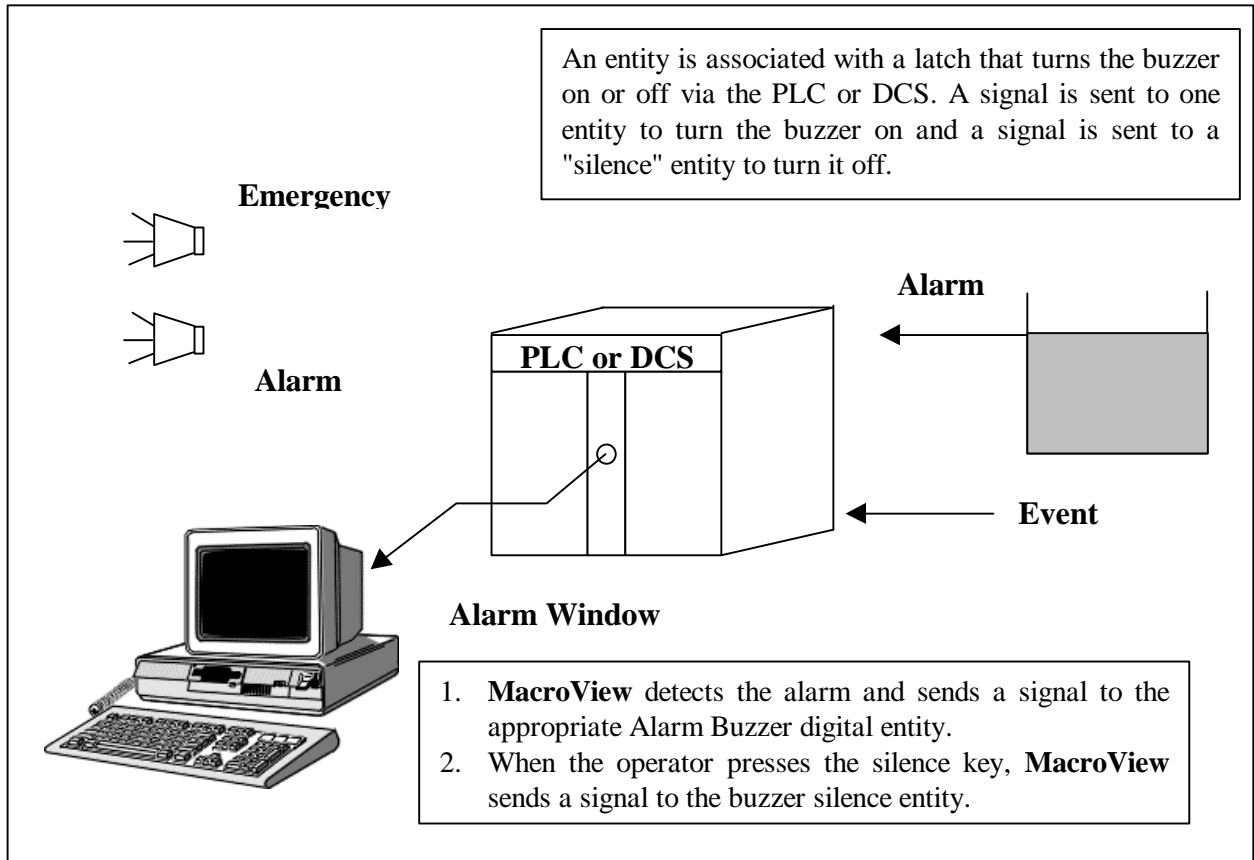
The alarms are printed on the system printer. The format of the printout is as shown in the diagram below.



If for any reason the printer fails, you can still retrieve the alarms from the [histmsgs.dbf](#) database in the [msgs](#) directory.

## How Alarm Buzzers are linked to the Keyboard

The diagram shows how *MacroView* controls the buzzers when there is an alarm occurrence



## How the Alarm Buzzers Work

There are 4 buzzers, each representing a different priority alarm and each being controlled by a different PLC or DCS entity. The operator can only silence and acknowledge alarms that are in areas of the process that are assigned to this console.

**Note:** *MacroView* only sends a signal to the entities controlling the buzzers and a reset signal to turn the buzzers off. The logic in the PLC (either ladder logic or sequence tables) is responsible for ensuring that the signal latches the buzzer on or off. Note: the signal is normally configured to be a pulse in PLC systems. In other systems the device needs to check the signal and reset it as needed.

The alarms application finds the entry in the `consoles.dbf` database for the server. Whenever an emergency alarm occurs, the entity in `BUZ_EM` is set to 1. The same is true for Alarms, Warnings, and Operator guides. A buzzer will not be set for System Alarms.

When an operator presses the silence key in the Navigator, any configured silence entities for his console and/or area will be set.

## 6.2 The Alarms Configuration Screen

To get to the alarms configuration screen, just select *Alarming:Detail* in the Configurator.

**Alarms Configuration Options**

The screenshot shows the 'Alarms Configuration Options' window. At the top is a menu bar with 'File', 'Data Sources', 'Data Types', 'Entities', 'Security', 'Historization', 'Displays', and 'Alarming'. Below the menu are several input fields for configuring an alarm: Entity (CSTCPMP), Attribute (ALM), Deadband (0.0), Alarm Type (Digital Scan), Color (Dark Green), Expression, Action, Message (Caustic pump trouble), Priority (Emergency), Desc., Group (2), Delay(s), Filter ON (F.O OFF), and Search. At the bottom is a table listing various alarms.

No.	Entity	Attribute	Alarm Type	Group	Color	Delay
1	CSTCPMP	ALM	DIGISCAN	2	3	1
2	AGITATOR	PV	3ISCAN	3	10	1
3	SCANT	DLH	DEVN	4	0	IF PVL >
4	FRANK			5	0	
5	WTRPMP	INTL		0	3	
6	DEMO1	DVPA	LOW	0	0	
7	FRANK			0	0	
8				0	0	
9	AGITATOR	DLH	DEVN	2	6	IF PVL >
10	CSTCPMP	DLH	DEVN	2	0	IF PVL >
11	WTRPMP	DEVN		2	0	IF DTI >

**Alarms Configuration Options**

1. **Menu:** Provides a means of Adding Blank Records, Adding Like (cloning), and Deleting Alarms.
2. **Detail Area:** The detail area shows the various items of the selected alarm in the almspec.dbf database. The next sections in this chapter provide detailed instructions on how to configure these items.
3. **Filter and Search:** The Browse widget can be filtered to show only those alarms that satisfy a certain condition. You create a filter by (i) selecting a field to filter on from the pull-down combo box and (ii) entering a criteria into the filter area. When you click on the filter button, only those records satisfying the filter conditions will be shown. You may also use the same filter to search through the database.
4. **Browse Area:** This shows a (possibly filtered) window of alarms. You may scroll through the alarms and examine details about the selected alarm simply by clicking on the record of interest.
5. **Message Area:** Suggestions, Help and Error Messages are sent to this area to assist you in your configuration.

## 6.3 Alarm Configuration

The next sections describe in detail the various fields you need to fill in to configure the alarms. The pages are arranged in the order that you are likely to enter the data.

Remember, you may assign multiple alarms to any attribute, however, please be aware of assigning too many alarms both in processor usage and operator loading.

**Note:** Once you have finished the configuration, remember to restart the alarms program by selecting *Alarming:Restart Alarm Manager*. This will force the Alarm Manager to read the changes you have made to the database. (A stopwatch appears followed by the message "All Configurator Views Reset".)

## Alarm Configuration (Entity & Attribute)

Select the entity and attribute with which this alarm is to be associated.

### Entity and Attribute

*What you type*

Enter the entity and select the relevant attribute you want to be alarmed. The attribute is generally (but not always) involved in the alarm calculation. (See Note below).

*Example*

**Entities:** FIC109, WATERPID etc.  
**Attributes:** PV, MV, SV, RUN\_HRS. You may alarm on any attribute from any source. You may have as many alarms as you need associated with any one attribute.

*How it Works*

**MacroView** generally uses this value with the **TYPE** and **EXPRESSION** fields to decide whether an alarm exists or not.

*Example*

If the **ATTRIBUTE** is **MV**, **TYPE** is **HIGH** and **EXPRESSION** is 80, then the alarm is activated when the **MV** exceeds 80.

In some cases, the alarm may be associated with more than one attribute (e.g. with Alarm Type EXPRN). In others, there is no strong relationship with any attribute (e.g. with REPORT alarms this could happen). In these cases, choose an attribute that has the most logical association

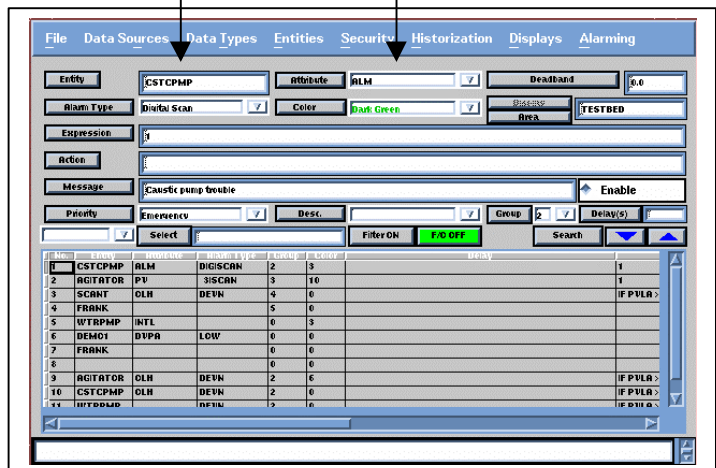
### 1 Alarming:Detail

This brings up the Alarm Detail screen.

*How to get there*

2 Click on the Alarm to be modified:  
 The Alarm detail will appear in the top window.

3 Alternatively, you may add a blank record using *Alarm:Detail:Add Blank* and edit the new record.



## Alarm Configuration (Type)

The alarm type specifies what kind of processing is performed. The format of the expression is largely dependent on the alarm type. Consult the table in the next section Alarm Type Configuration.

### Type

*What you type*

Select the type of alarm you require from the pull down combo. For more information on the types, please look at the table in the next section. The valid options include:

<b>HIGH HIHI</b>	If the attribute value exceeds the expression	<b>EXPRN</b>	Alarms if the free format expression evaluates to True or 1.
<b>LOW LOLO</b>	If the attribute value goes below the expression.	<b>DIGISCAN</b>	Alarms if the attribute equals the expression
<b>DEVN</b>	Alarms if the target (specified in the expression) deviates from the attribute by the amount also specified in the expression).	<b>REPORT</b>	Alarms if the report-by-exception string is received by the alarm manager (See Separate examples after the table).
		<b>CHANGE</b>	Alarms if there is a change to the attribute.
<i>Note</i>	<i>Deadbands are used for the alarms in this column.</i>	<i>Note</i>	<i>Deadbands are not used for the alarms in this column</i>

*Example*

If you wanted a HIGH alarm plus a HIHI alarm, you would have two entries associated with these two alarms for that entity. **MacroView** uses the alarm type to decide how to handle the EXPRESSION field. For example, if the type is LOW then **MacroView** uses the value in the EXPRESSION field as the low threshold.

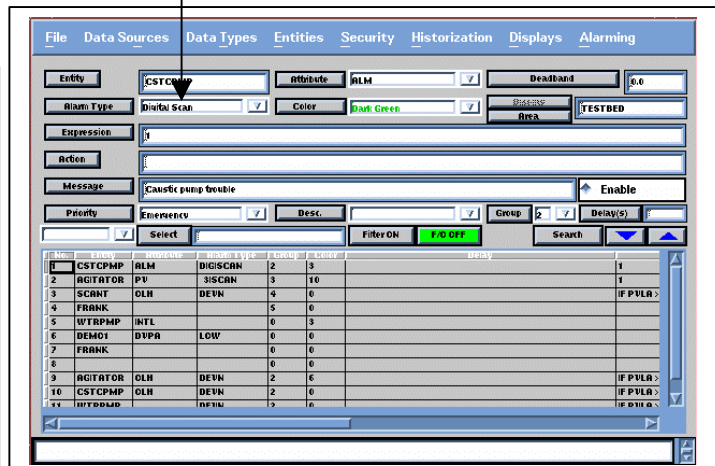
*How it Works*

If the type is EXPRN, then **MacroView** expects to find a free format expression in the EXPRESSION field. If you set the type to REPORT, the processing will be done in the PLC or DCS and **MacroView** will only report the alarm. How this is done is specific

- 1 **Alarming:Detail**  
This brings up the Alarm Detail screen.

*How to get there*

- 2 Click on the Alarm to be modified:  
The Alarm detail will appear in the top window.
- 3 Alternatively, you may add a blank record using **Alarm:Detail:Add Blank** and edit the new record.



## Alarm Configuration (Expression)

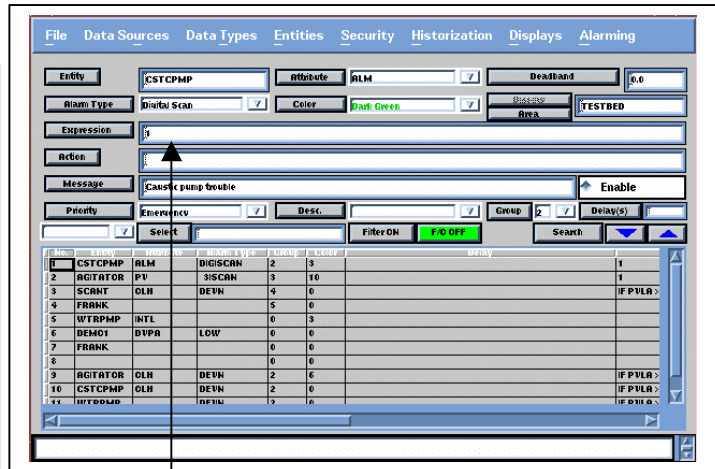
The expression is used in conjunction with the alarm type to determine if an alarm exists.

### 1 Alarming:Detail

This brings up the Alarm Detail screen.

*How to get there*

- Click on the Alarm to be modified: The Alarm detail will appear in the top window.
- Alternatively, you may add a blank record using *Alarm:Detail:Add Blank* and edit the new record.



### Expression

*What you type*

Enter the expression - it can be a number e.g. 80. It can be an attribute e.g. PH or it can be a free format expression e.g. LS(\$)= "MAN". You must limit the size of the expression to 40 characters.

*Hint*

If the PLC or DCS has attributes associated with the alarm settings e.g. PH for process high, try to use these instead of fixed values. This avoids having to set the values in the PLC or DCS and again in **MacroView**.

*How it Works*

The valid codes used in the expression are described in the table on the next page. For more information on expressions, please read the section "Expressions in Alarms" in the Programs Chapter.

**Note:** This version of alarms can not handle the format entity.attr, only the format attr(entity) may be used.

*Example*

ATTRIBUTE	TYPE	EXPRESSION	MEANING
SV	LOW	33	Alarm if the SV goes below 33.
SV	LOW	PVL	Alarm if the SV goes below the PVL attribute.
PV	EXPRN	LS(\$)="MAN" .AND. PV(\$)>PH(\$)	Alarm if in manual AND the PV is greater than the high level.

## Alarm Type Configuration

This table illustrates the different types of alarm processing available.

**Note:** that examples are displayed in Italics.

**Table 4: Alarm Type Configuration**

Type	Expression	Dead Band Used	Attribute	Description
HIGH	A constant (e.g. 95) or an attribute name. <i>PH</i>	Yes	<i>SV</i>	Alarm if attribute exceeds constant or attribute specified in the expression field.  <i>High alarm if <math>SV &gt; PH</math></i>
HIHI	A constant (e.g. 99) or an attribute name. <i>HH</i>	Yes	<i>PV</i>	Alarms if attribute exceeds the constant or attribute specified in Expression.  <i>HIHI alarm if <math>PV &gt; HH</math></i>
LOW	A constant (e.g. 10) or an attribute name. <i>ML</i>	Yes	<i>MV</i>	A low alarm is produced if the attribute goes below the constant or attribute in the expression held.  <i>Low alarm if <math>MV &lt; ML</math></i>
LOLO	A constant (e.g. 5.5) or an attribute name (e.g. LL) <i>5.5</i>	Yes	<i>PV</i>	A Low alarm is produced if the attribute goes below the constant or attribute in the expression held.  <i>LOLO Alarm if <math>PV &lt; 5.5</math></i>
DEVN	Two fields separated by a comma i.e. target, devn_limit. Either field can be a constant or an attribute.  <i>SV, DL</i>	Yes	<i>PV</i>	Alarms if attribute deviates more than the deviation limit away from the target.  <i>DEVN Alarm if <math>(PV-SV) &gt; DL</math></i>
EXPRN	A free form expression. You must use a \$ to represent the current entity name.	No		Alarms if the expression evaluates to TRUE or 1. You may use any valid expression, (See the Programs Chapter.) with any entity. You are limited to 48

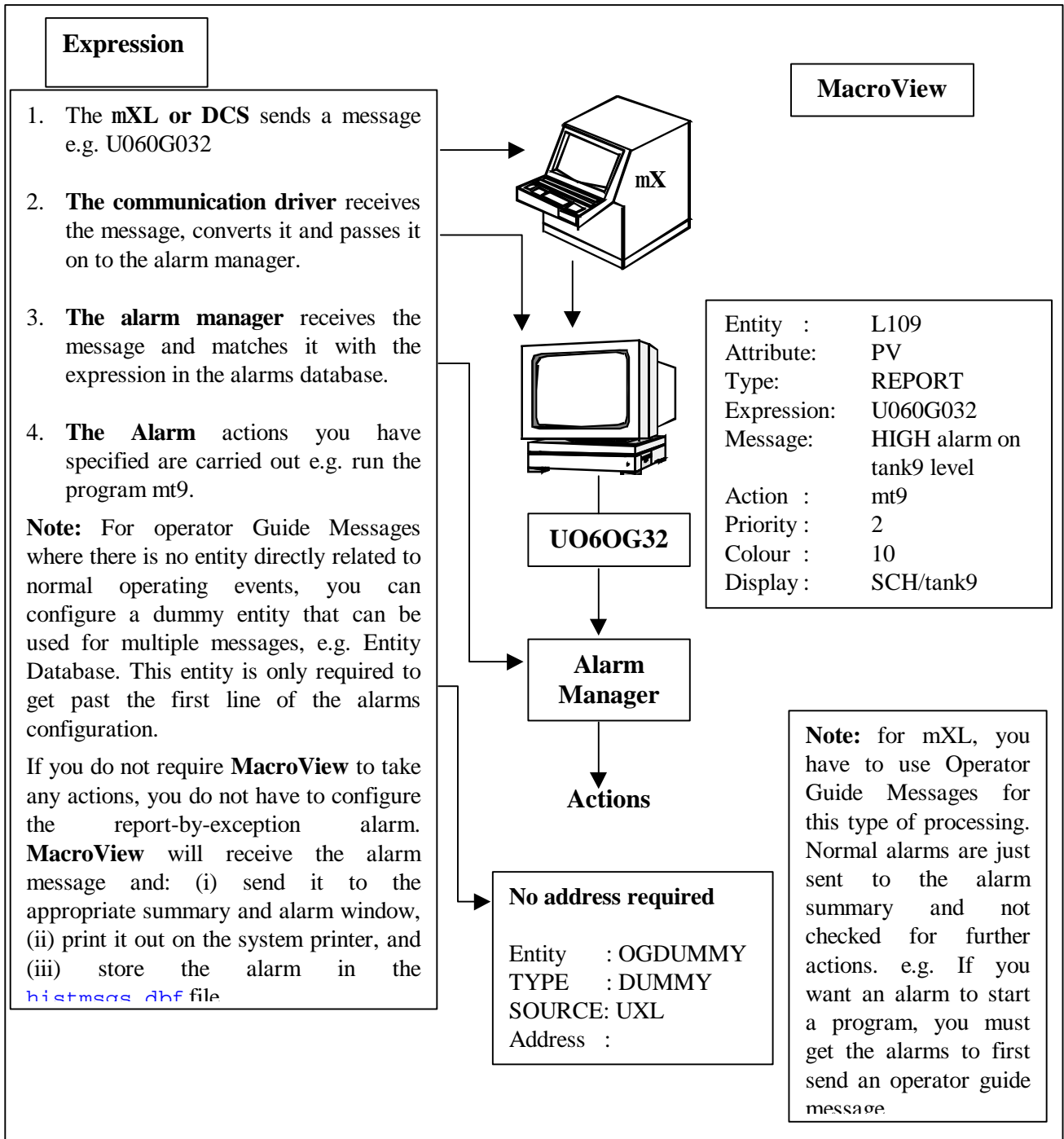


## Report by Exception Alarms

With the report-by-exception alarms, all the processing is done in the external equipment. **MacroView** receives a report that an alarm has occurred and the normal alarm actions are initiated. The format of the message is communication driver specific.

See the driver documentation for this specific information.

The alarm manager tries to match the message received in the report message with the **EXPRESSION**. If it can match them, the configured alarm actions are triggered. If not, the message is just sent to the alarm summary.



## Alarm Configuration (Deadband)

This page describes what to enter in the Deadband (Alarm Hysteresis) field in the alarm configuration.

### Expression

*What you type*

Type in the percentage (of full scale) that defines the alarm deadband e.g. 1 for 1% (The allowable range is 0.0 - 9.9)

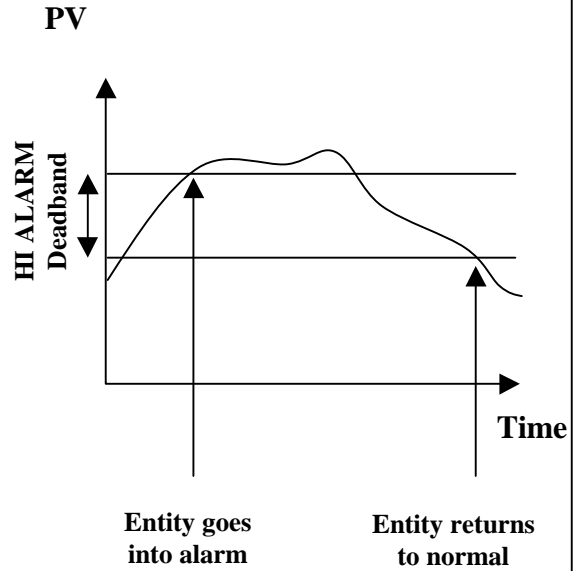
*How it Works*

Alarms return to normal only when they go outside the deadband.

*Hint*

Use the deadband to stop entities continuously drifting in and out of alarm when the value is close to the alarm threshold.

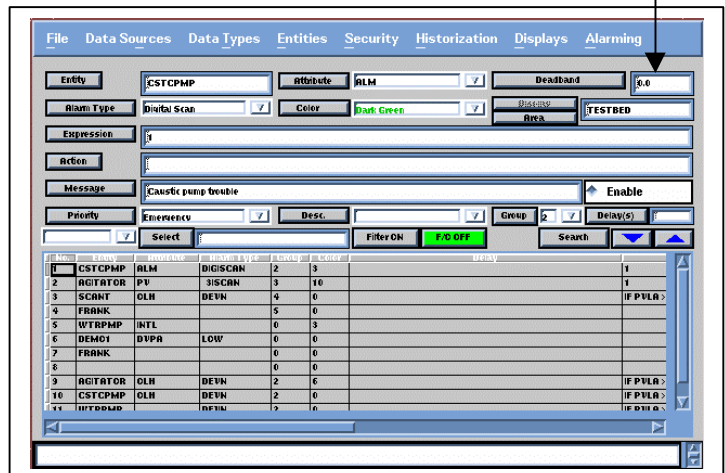
The Deadband is only applicable for the following types of alarm: [HIGH](#), [LOW](#),



- 1 **Alarming:Detail**  
This brings up the Alarm Detail screen.

*How to get there*

- 2 Click on the Alarm to be modified:  
The Alarm detail will appear in the top window.
- 3 Alternatively, you may add a blank record using **Alarm:Detail:Add Blank** and edit the new record.



## Alarm Configuration (Priority)

This page describes what to enter in the Priority field in the alarm configuration.

### Priority

*What you type*

Select the priority of the alarm, i.e. which alarm summary and the priority in that alarm summary. This can have the following values:

- 0 - System Alarm                      System Alarm Summary
- 1 - Emergency                      } Alarm Summary
- 2 - Alarm                              }
- 3 - Warning                            }
- 4 - Operator Guide                  Operator Guide Message Summary
- 5 - Log only                           Available only in the historical messages summary

*Example*

If the HIGH alarm is urgent and the HIHI alarm is an emergency, you would put the HIGH priority at 2 and the HIHI priority at 1.

*How it Works*

Each priority is associated with an individual buzzer - thus high priority alarms should have a loud buzzer and warning only alarms should have a muted buzzer. You configure each buzzer to a digital entity in the CONSOLE option in the configurator menu. (You can of course decide to have only one buzzer or no buzzers at all.) See the section earlier on "How alarms are linked to the keyboard". The priority of the alarm is also stored in the [histmsgs.dbf](#) file for later analysis.

*Hint*

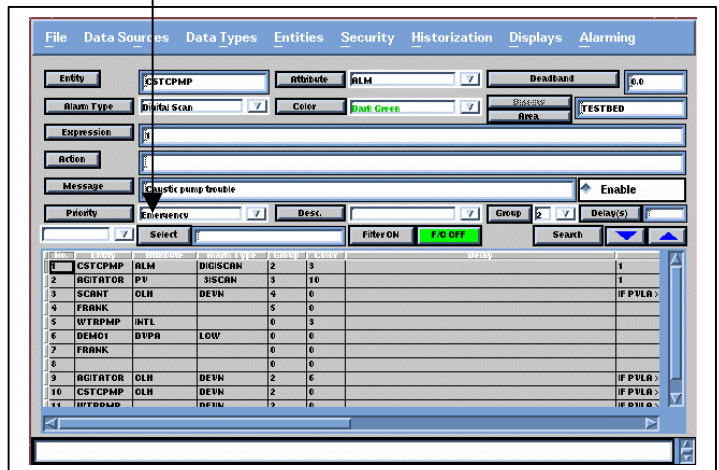
Decide on an alarm strategy before you start configuration. A general rule; keep the number of alarms down, only use the emergency priority when it really is an emergency.

### 1 Alarming:Detail

This brings up the Alarm Detail screen.

*How to get there*

- 2 Click on the Alarm to be modified: The Alarm detail will appear in the top window.
- 3 Alternatively, you may add a blank record using *Alarm:Detail:Add Blank* and edit the new record.

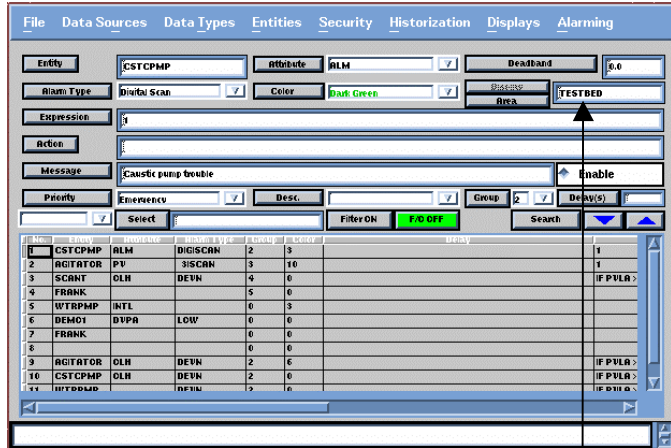




# Alarm Configuration (Display)

The Display field may be used as a simple link to the relevant display.

- 1 **Alarming:Detail**  
This brings up the Alarm Detail screen. *How to get there*
- 2 Click on the Alarm to be modified:  
The Alarm detail will appear in the top window.
- 3 Alternatively, you may add a blank record using **Alarm:Detail:Add Blank** and edit the new record.



## Display : SuperFind Option

*What you type*

Enter the screen name to be linked to this alarm.

*Things to Note*

If you do not enter a display name here, the display will default to the detail display of this entity. Define the Superfind Display screen as follows:

- (i) By screen name, i.e. apl/screen\_name, e.g. SCH/boiler\_9 (you can use up to ten characters for the screen\_name which is the name of the graphic metafile) or,
- (ii) By screen number, i.e. aplnnn, e.g. SCH0004 - Where nnnn is the screen number and apl is one of the following application codes:

- |                 |                     |
|-----------------|---------------------|
| SCH = Schematic | SYS = System Alarm  |
| OVW = Overview  | MSG = Message       |
| GRP = group     | TRD = Trend         |
| HLP = Help      | DTL = <entity name> |
| ALM = Alarm     |                     |

## Alarm Configuration (Message and Colour)

This page describes how you specify the alarm message and the colour or the message as it appears in the alarm summary.

### Message, Colour

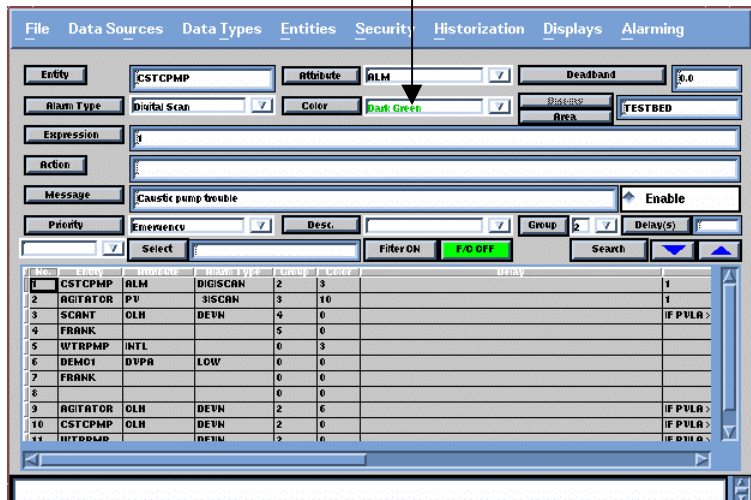
*What you type*

Enter the message you want displayed on the alarm summary (up to 32 characters) and the colour of the message on the screen. You can choose a specific colour code or choose a colour that is related to the priority. If you want the colour to be priority-based in the pull down combo and the standard priority based colours will appear. If you want a specific colour, just enter the colour in the pull-down combo. You can set the colours associated with priorities 1,2 and 3 in the \$MACRODIR/./msgs/app1/msgs.ini text file. Just call up the alarms, type p, then enter config, and select the defaults soft key.

*Things to Note*

Typical priority driven colours include: Emergency: Red, Alarm: Half Yellow, Warning: Yellow Message: Colour depends on message e.g. cyan, white.

**Note:** *Acknowledged* alarms (if the operator chooses to view them) will be inverted. Alarms that *have returned to normal but have not been acknowledged* will show with their text as Dark Blue. Acknowledged Alarms *that have returned to Normal* will be automatically deleted. You can choose to view RETURN TO NORMAL messages (green background) by starting the *mvmsgs* program with the -l option.



## Alarm Configuration (Action) (UNIX System)

You can initiate a program when an alarm occurs by entering the program name in the action field as defined below.

### Actions

*What you type*

You type in the name of the program you want executed when the alarm occurs. The program must be executable (use `chmod a+x <programe>`) and it must be in the `$HOME/bin` directory or have a path relative to the `$HOME/bin` directory.

*Example*

<code>mt9</code>	A program in the bin directory to empty a tank.
<code>../rep/sitrep</code>	A program in the rep directory to print a "situation report".
<code>mshell</code> <code>bat_end.ms</code>	A program that runs a meta script called ( <code>bat_end.ms</code> ) on completion of a batch.
<code>generic</code>	A custom generic alarm handler that performs certain tasks for all alarms.

*Things to Note*

The program will inherit the Unix shell and the user of the alarms program. Typically this will be the Bourne shell and the super-user (root). When the program is started, it is passed a number of variables as arguments, these are described in the next section.

**Note:** Examples on writing programs in Shell script, dBase, etc., can be found in the Programs Chapter.

Please take special note of how to run the programs in background.

*How it Works*

The next page shows two simple examples of executable scripts.

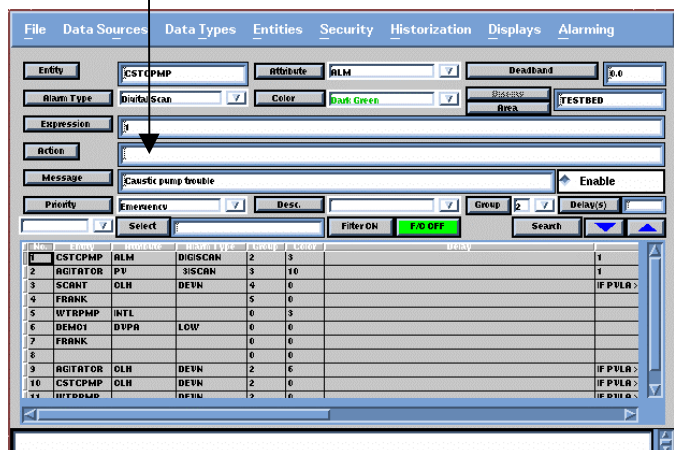
### 1 Alarming:Detail

This brings up the Alarm Detail screen.

*How to get there*

2 Click on the Alarm to be modified: The Alarm detail will appear in the top window.

3 Alternatively, you may add a blank record using **Alarm:Detail:Add Blank** and edit the new record.



## Alarm Configuration (Action) (NT System)

You can initiate a program when an alarm occurs by entering the program name in the action field as defined below.

### Actions

*What you type*

You type in the name of the program you want executed when the alarm occurs. The program must be created as a batch file and it must be in the %HOME%/bin directory or have a path relative to the %HOME%/bin directory.

*Example*

mt9.bat	A program in the bin directory to empty a tank.
..\rep\sitrep.bat	A program in the rep directory to print a "situation report".
mshell bat_end.ms	A program that runs a meta script called (bat_end.ms) on completion of a batch.
generic	A custom generic alarm handler that performs certain tasks for all alarms.

*Things to Note*

The program will inherit the Unix shell and the user of the alarms program. Typically this will be the Bourne shell and the super-user (root). When the program is started, it is passed the following variables as arguments.  
**Note:** Examples on writing batch files can be found in the Programs Chapter, while the use of the meta script language is covered in the Meta Script Chapter.

*How it Works*

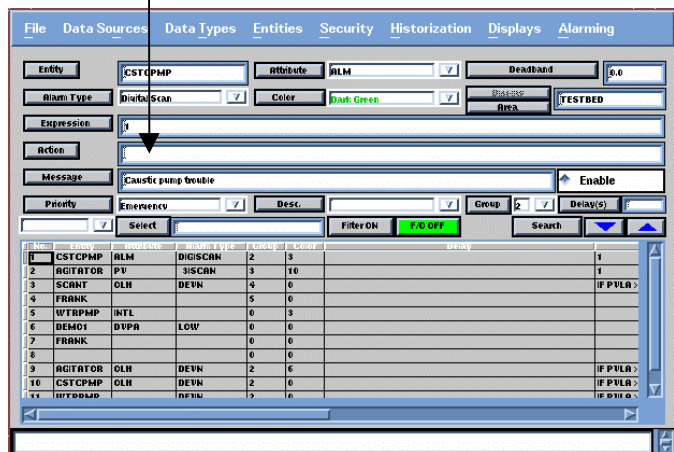
The next page shows two simple examples of executable scripts.

**1 Alarming:Detail**

This brings up the Alarm Detail screen.

*How to get there*

- Click on the Alarm to be modified: The Alarm detail will appear in the top window.
- Alternatively, you may add a blank record using **Alarm:Detail:Add Blank** and edit the new record.



## 6.4 Arguments passed to the action program (UNIX System)

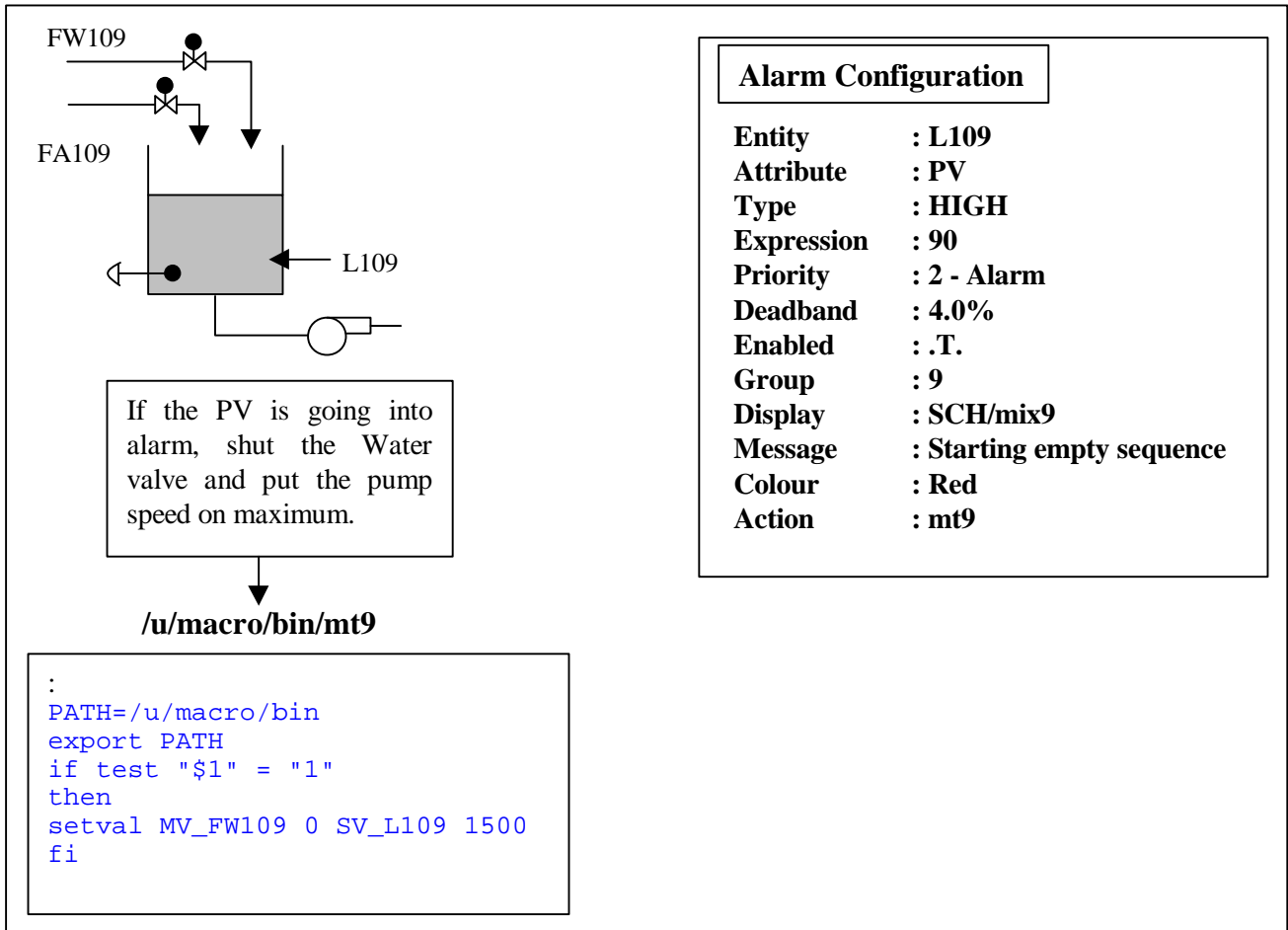
The following table lists the arguments or parameters, which can be passed to a UNIX script, when it is executed from the Action field of the Alarm Specification database. The example program, *mt9*, on the following page, shows the use of \$1 as a parameter being passed to the script. In this example it is assumed that no other parameter was passed to the script in the command line and therefore \$1 refers to a condition going INTO or OUT OF alarm (as per the table below). To further explain this, let us look at an example where the Action field command line reads *mt9 vall*, where *vall* is being passed to the *mt9* script. In UNIX script syntax, \$1 is the first parameter being passed to the *mt9* script and in this example will refer to *vall*. Therefore, **it is important to note** that the argument numbers in the table below will need to be incremented by the number of arguments used in the command line of the Action field. For example, where I have used *vall* in the command line, I would have to use \$2 to refer to the INTO and OUT OF alarm condition in the *mt9* script.

**Table 5: Arguments to be passed to a UNIX script**

Argument	Description	e.g.
\$1	1 If going into alarm. 0 if returning to normal	1
\$2	The entity name	L109
\$3	The attribute name	PV
\$4	The current value of the attribute	84.9
\$5	The display colour	10
\$6	The area the entity is associated with.	Power
\$7	The alarm priority	2
\$8	The alarm group	9
\$9	The associated display	SCH/mix9

## Action Example (UNIX System)

Simple example:



## Generic Example (UNIX System)

You can write a single script that can react to all alarms. The generic program reads the arguments sent by the alarm manager and performs different tasks dependant on the values of the arguments.

This simple program can be used as a "template" for more complex programs; for example a program that sends paging messages to different people for different areas of the process.

For the generic type programs, just enter the name of the generic program (with any optional arguments) in the action field of all alarms that you want this processing carried out on. The following example script is an example of a generic alarm script that simply reports the details of each alarm that occurs.

**Note:** The example is one in which no arguments have been used in the calling program.

```

:
# This program simply shows how the arguments sent by the alarm
# manager may be used
# in the action program
if test "$1" ="1"

```

```
then
echo $3 \ ($2\) is in alarm.
else
echo $3 \ ($2\) is not in alarm.
fi
echo Its current value is $4.
echo Its configured alarm display colour is $5.
if test "$6" = ""
then
echo It is in area $6.
fi
echo The alarm is of priority $7.
echo the alarm is in alarm group $8.
echo Its superfind display is $9.
#
# The Bourne shell supports only $0 through to $9.
# To get access to the 10th parameter we need to do a shift.
#
shift
echo The first associated alarm argument is $9 echo
```

## 6.5 Arguments passed to the action program (NT System)

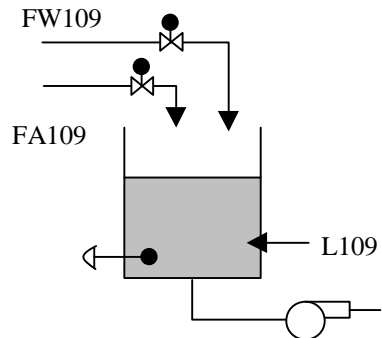
The following table lists the arguments or parameters, which can be passed to a batch file, when it is executed from the Action field of the Alarm Specification database. The example program, *mt9.bat*, on the following page, shows the use of **%1%** as a parameter being passed to the batch file. In this example it is assumed that no other parameter was passed to the batch file in the command line and therefore **%1%** refers to a condition going INTO or OUT OF alarm (as per the table below). To further explain this, let us look at an example where the Action field command line reads *mt9 vall*, where *vall* is being passed to the *mt9.bat* batch file. In batch file syntax, **%1%** is the first parameter being passed to the *mt9.bat* batch file and in this example will refer to *vall*. Therefore, **it is important to note** that the argument numbers in the table below will need to be incremented by the number of arguments used in the command line of the Action field. For example, where I have used *vall* in the command line, I would have to use **%2%** to refer to the INTO and OUT OF alarm condition in the *mt9.bat* batch file.

**Table 6: Arguments to be passed to a NT script**

Argument	Description	e.g.
<b>%1%</b>	1 If going into alarm. 0 if returning to normal	1
<b>%2%</b>	The entity name	L109
<b>%3%</b>	The attribute name	PV
<b>%4%</b>	The current value of the attribute	84.9
<b>%5%</b>	The display colour	10
<b>%6%</b>	The area the entity is associated with.	Power
<b>%7%</b>	The alarm priority	2
<b>%8%</b>	The alarm group	9
<b>%9%</b>	The associated display	SCH/mix9

## Action Example (NT System)

Simple example:



If the PV is going into alarm, shut the Water valve and put the pump speed on maximum.

**C:\Users\Macro\Bin\mt9.bat**

```

PATH=
C:\Users\Macro\Bin\mt9.bat
Set Path =
if "%1%" = "1"
then
setval MV_FW109 0 SV_L109 1500
end
                
```

Alarm Configuration	
Entity	: L109
Attribute	: PV
Type	: HIGH
Expression	: 90
Priority	: 2 - Alarm
Deadband	: 4.0%
Enabled	: .T.
Group	: 9
Display	: SCH/mix9
Message	: Starting empty sequence
Colour	: Red
Action	: mt9

## Generic Example (NT System)

You can write a single script that can react to all alarms. The generic program reads the arguments sent by the alarm manager and performs different tasks dependant on the values of the arguments.

This simple program can be used as a "template" for more complex programs; for example a program that sends paging messages to different people for different areas of the process.

For the generic type programs, just enter the name of the generic program (with any optional arguments) in the action field of all alarms that you want this processing carried out on. The following example script is an example of a generic alarm script that simply reports the details of each alarm that occurs.

**Note:** The example is one in which no arguments have been used in the calling program.

```

:
# This program simply shows how the arguments sent by the alarm
# manager may be used in the action program
    
```

```
if "%1%" = "1" echo %3% \(%2\) is in alarm
if not "%1%" = "1" echo %3% \(%2\) is in not alarm

echo Its current value is %4%
echo Its configured alarm display colour is %5%
if test "%6%" = "" if "%6%" = "" echo is in area %6%

echo The alarm is of priority %7%
echo the alarm is in alarm group %8%
echo Its superfind display is %9%
echo The first associated alarm argument is %9%
echo
```

## 6.6 Checking out the Alarm Configuration

We recommend you follow the procedure below to check out the alarms.














- (i) Get a summary printout of all the alarms in the system.
- (ii) Force the alarm on and verify all the actions you expect to occur do occur.
- (iii) If the alarm does not behave, as it should, then reconfigure it, restart the alarm manager and go back to (i) above.

## 6.7 Navigator Alarms Colours

When viewing the alarms in the Navigator's alarms application, the following colours are used.

**Note:** This assumes that the alarms' colours have been configured as priority based.

**Table 7: Default Alarm Colours in the Navigator**

Priority	Alarm Status	Colour
Emergency	In alarm, un-acknowledged	Black on Red 
Emergency	Out of alarm, un-acknowledged	Dark Blue on Red 
Emergency	In alarm, acknowledged	Red on Black 
Alarm	In alarm, un-acknowledged	Black on Orange 
Alarm	Out of alarm, un-acknowledged	Dark Blue on Orange 
Alarm	In alarm, acknowledged	Orange on Black 
Warning	In alarm, un-acknowledged	Black on Yellow 
Warning	Out of alarm, un-acknowledged	Dark Blue on Yellow 
Warning	In alarm, acknowledged	Yellow on Black 
System Alarm	Un-acknowledged	Black on Blue 
System Alarm	Acknowledged	Blue on Black 
Operator Guides	Un-acknowledged	Black on Green 
Operator Guides	Acknowledged	Green on Black 
All	OUT OF ALARM message	Black on Light Green 