Chapter 20

ALARMS

Edition 3

Overview

Overview

A suite of alarm function blocks has been added to the PC3000 Function Block library. These provide a means of logging exception conditions and events. The blocks provide for multiple alarm/event sources and record time alarms occurred, value at which they occurred together with an optional text message for each entry. The PC3000 Real Time Clock is used to time stamp each alarm entry. The alarm system is based around two principal types of blocks.

Alarm Sensors - these are blocks with a Boolean input which register the alarm state. The actual conditions which generated the alarm is defined external to the block by means of soft wiring in the user programme. This permits maximum flexibility in defining the source of the alarm or event. A simple example might be to test for Loop_1.PV> Max Temp. When the alarm state changes, the state is passed to an associated alarm buffer block.

Alarm Buffers - these are blocks which store or log information associated with each alarm. There are two types provided:

Cur_Alms	Logs data associated with current or 'live' alarms. This block provides information about the STATE of an alarm.
History	Logs historical i.e. cleared alarms for batch traceability or process start up 'post-mortem'.

There are four possible states for an alarm entry in the current alarm buffer:

Alarm State	Description
Active, Not Acknowledged	The alarm sensor has detected an alarm condition. The supervisor/operator interface has taken no action
Active, Acknowledged	The alarm sensor has detected an alarm condition. The supervisor/operator interface has tacknowledged the alarm but the condition still exists.
Not Active, Not Acknowledged	The alarm sensor has detected an alarm condition. The supervisor/operator interface did not tacknowledge the alarm but the alarm condition no longer exists. ('fleeting' or missed alarm)
Not Active, Acknowledged	The alarm sensor has detected an alarm condition. The supervisor/operator interface has acknowledge the alarm; the alarm condition has been CLEARED. In this state the alarm is transferred from the live or current alarm buffer and becomes a historic alarm in the history buffer.

Table 20-1 Alarm Entry States

The sensor blocks provide 23 input parameters which may be used to identify the alarm/event source. These parameters are described as Type and Area. The parameters may be used either:

1. Where alarms must be grouped i.e. all barrel zones on an extruder, the Area parameter may be used to classify any alarm from this part of the machine. If the different alarms can occur e.g. an over temperature alarm and a setpoint deviation alarm then further identification is possible by using the Type parameter. The over temperature alarm could be described as Type 1 and the deviation alarm as Type 2.

2. Where it is not required to group alarms, the combination of Type and Area are used to create a unique alarm number for each alarm.

   Note: No two alarms can have the same Type and Area as it will not be possible to establish where the alarm originated.

In addition to the sensor block, a detector block provides standard alarm sensing strategies for absolute, deviation alarms and rate of change. The block monitors an input variable e.g. PV and compares it with setpoint and limit inputs. The limits detected are high alarm, low alarm, deviation high alarm, deviation low alarm, deviation band alarm and rate of change alarm.

In Figure 20-1 the analogue input (either a monitor or control input) has been wired to a Detector function block. This compares the analogue input value with the alarm threshold of 650 and if exceeded will result in the Hi_Alarm output being set. This is sensed by the alarm sensor wired to the detector and time stamped. The sensor has a configuration parameter Bufld which associates the sensor with the current alarm buffer that shares the same ID. The Cur_Alms block also includes a configuration input, LogId, which is used to associate the History buffer into which alarms will be entered when cleared.

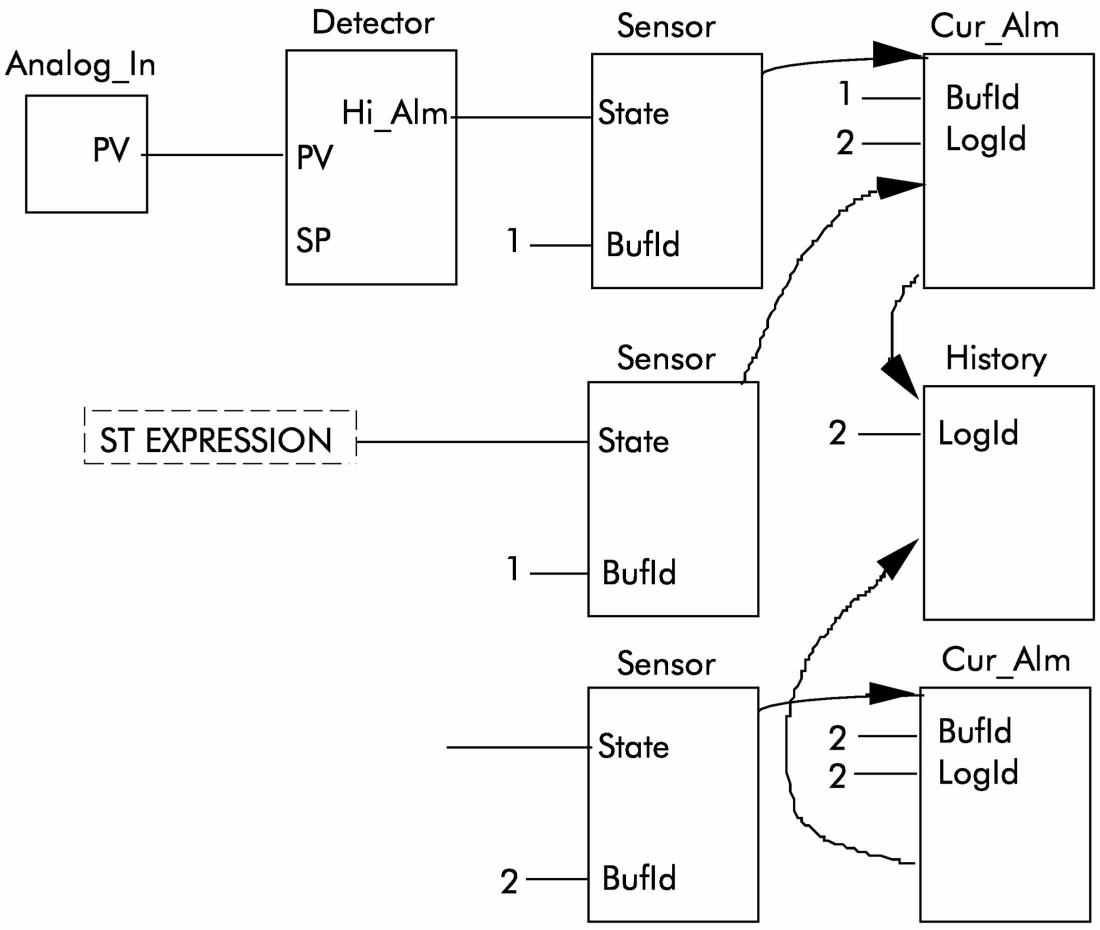


Figure 20-1 Overview of the alarm system

The diagram also shows how an alarm sensor block may be used in conjunction with a Structured Text wiring statement to implement non-standard alarm strategies. An example might be:

```
Sensor.State:- tcl.PV >= 250 and BatchNo.Val = 10 AND
ProcTime.Elapsed-Time > T#5h25m;
```

Finally, the diagram shows a third sensor with a different Bufld. This allows alarms, perhaps from a different process to be associated with a separate alarm buffer. In this case the buffer is identified with a Bufld of 2.

CUR_ALMS FUNCTION BLOCK

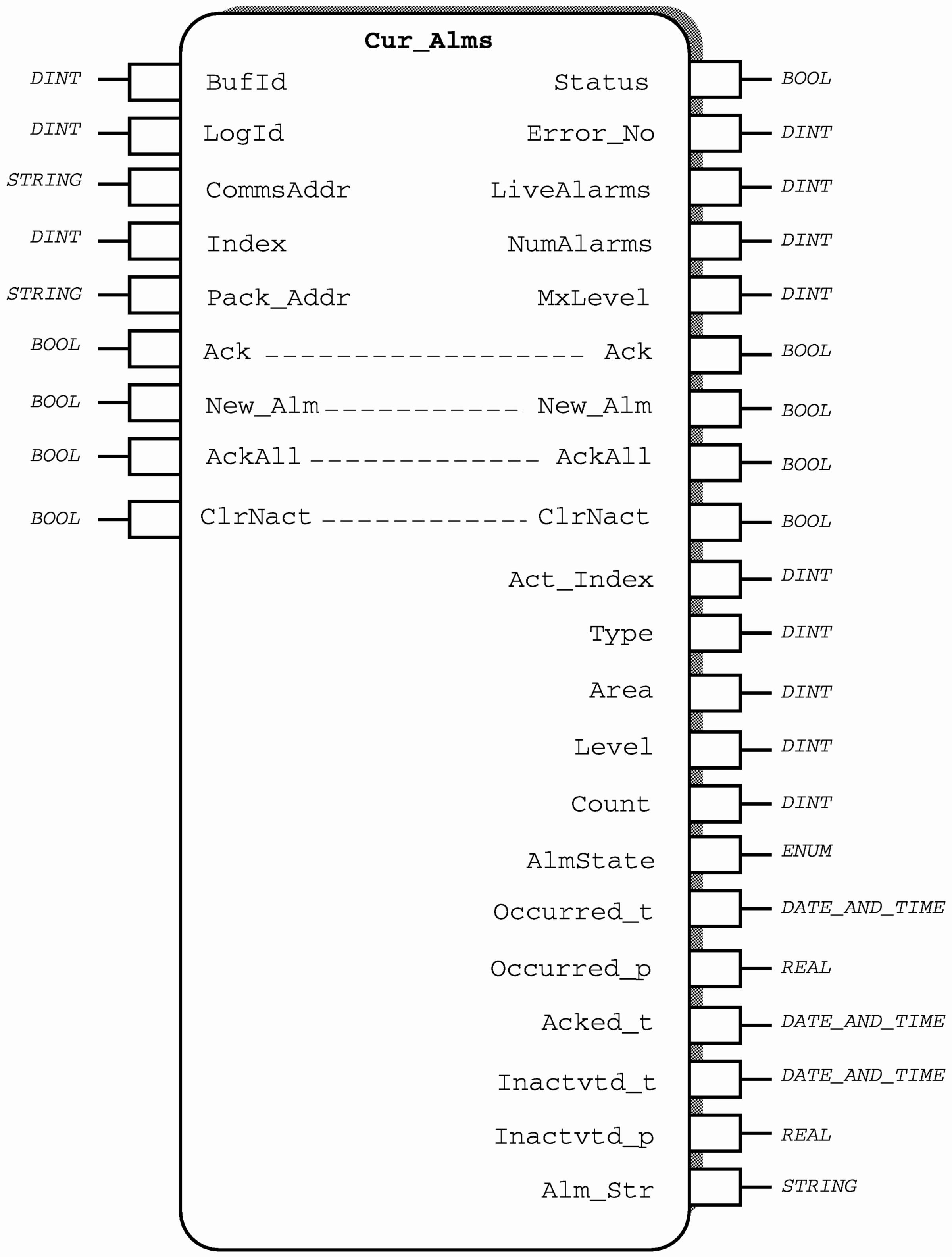


Figure 20-2 Cur_Alms Function Block

Functional Description

The Cur_Alms function block contains an array of currently uncleared or live alarms and allows access to the elements of this array by means of an index input. It is intended for use with alarm sensor function blocks to provide a system for recording alarms and user defined events within the PC3000 controller.

The block also contains two internal communications slave variables. One provides a Eurotherm Bisync interface (when associated with a slave comms driver) and is used to access the complete data held in the function block. This would be used with supervisory computers such as ESP. The other slave variable provides a MODBUS/JBUS interface and allows access to a packed string containing alarm type and state information only. The latter is intended to provide simple access to key data and is intended primarily for use with the 9" and 12" terminals.

Alarms can be acknowledged via either of these tow slave parameters, or via the parameters of the function block itself. This allows supervisory systems, Xycom panels, and the Euro_Panel to be used as an interface to the alarm handling system.

The current alarm buffer is 'associated' with a number of alarm sensors (function blocks which provide the data about the alarm condition) by means of a buffer identifier. This allows the use of more than one live alarm buffer in the PC3000 Alarms from different parts of the machine or process could be associated with their own current alarm buffer.

The current alarm buffer also provides a mechanism for interfacing to History, historical alarm buffer. This buffer stores 'cleared' alarms. A History buffer is associated with a current alarm buffer by means of a log identifier.

Function Block Attributes

Type:	6670
Class:	ALARMS
Default Task:	Task_1
Short List:	BufID, LogID, NumAlarms, MxLevel
Memory Requirements:	6888 Bytes

Parameter Descriptions

The Cur-Alms function block provides the following parameters:

BufId (BID)

This is the alarm buffers identifier (ID). The ID must be unique. It is the reference by which alarm sensors are connected to or associated with the Cur_Alms function block.

LogId (LID)

This is the parameter which is used to associate a History function block with the current alarm buffer. Cleared alarms are passed to the History block with the log identifier (LogID) which matches this value.

CommsAddr (ADD)

This is the comms address for the multi-element or composite salve parameter. This address provides access via the Eurotherm Bisync protocol to the information contained in the alarm array. The date may be accessed as a single multi-element parameter, or as a number of separate elements. Reading the data at the slave address will return the multi-element parameter. Access to subsequent addresses will return only one element. it is not possible to specify data format for the individual elements and the default formats is assumed.

The data contained within the slave parameter takes the following form:

Offset	Parameter	Type
0	Composite parameter	
1	Index	Integer
2	Type	Integer
3	Area	Integer
4	State	Integer
5	Level	Integer
6	Count	Integer
7	Occured Time	Integer
8	Occured Position	Real
9	Acked Time	Integer
10	Inactivated Time	Integer
11	Inactivated Position	Real

The offset refers to the offset from the base address. As an example, if the slave address is set up with an address of 'EBA90' and the associated Bisync slave block has a GID setting of '0' the following address would apply:

Offset	Parameter	Type
00 A90	Composite parameter	
00 A91	Index	Integer
00 A92	Type	Integer
00 A93	Area	Integer
00 A94	State	Integer
00 A95	Level	Integer
00 A96	Count	Integer
00 A97	Occured Time	Integer
00 A98	Occured Position	Real
00A99	Acked Time	Integer
00 A9	Inactivated Time	Integer
00A9	Inactivated Position	Real

For further details on alarm addressing see the section covering alarm communications.

Index (1)

This parameter provides a means of accessing data stored in the alarm array.The function block outputs will display the alarm entry at position Index if it exists. As an example, if the tenth alarm was to be selected, the index would be set to 9 and the function block outputs would indicate data relating to this alarm.

Pack_Addr (PA)

This is the comms address for the packed slave parameter. This parameter is provided to emulate several applications which utilise the 9" and 12" Xycom terminals for alarm display and acknowledgements. If used with these terminals the address would be set for JBus use, i.e. 'JB****'.

For further details on alarm addressing see the section covering alarm communications.

Ack (ACK)

This is a boolean input allowing the currently selected alarm to be acknowledged. It is set by user and automatically cleared by block.

New_Alm (NAL)

This is a boolean status parameter which indicates that a new alarm has occurred. It is set by the block and should be cleared by the user. This parameter could be used to force an alarm screen to be displayed on the supervisory system or panel when a new alarm is received.

Status (ST)

This indicates the function block status. In normal operation the parameter will indicate the Go state. However, in the event of an error being detected the parameter will revert to the NOGO state and the reason for the error will be indicated by the Error_No parameter.

Error_No (ERR)

This indicates the function block error number. If an error is detected when the PC3000 program runs, the error will be indicated as shown in the table below and the block status will become NOGO.

Error Number	Error	Cause and Action
105	Too many buffers	There are more than 20m buffers (Cur_Alarms AND History function blocks). Any buffer indicating this error will not be operational and the number of buffers will be reduced.
106	Buffer ID out of range	The Log ID is less than 0 or greater than 9. Change the Buffer ID and sensor ID's associated with this buffer.
107	Duplicate ID	A buffer already exists with this ID. Change the buffer ID and sensor ID's associated with this buffer.

Table 20-2 Errors may be Detected and Indicated

Live Alarms (LA)

This indicates the total number of unacknowledged alarms in the alarm buffer

NumAlarms (NA)

This parameter indicates the total number of alarms in the buffer.

MxLevel (MXL)

This indicates the highest level alarm in the buffer i.e. the highest priority alarm.

Act_Index (AI)

This will be equal to Index if the parameter Index lies in the following range:

0<Index<=NumLogged

It will be equal to zero for values of Index outside this range.

Type (TYP)

This indicates the alarm type at the position in the log specified by Act_Index.

Area (A)

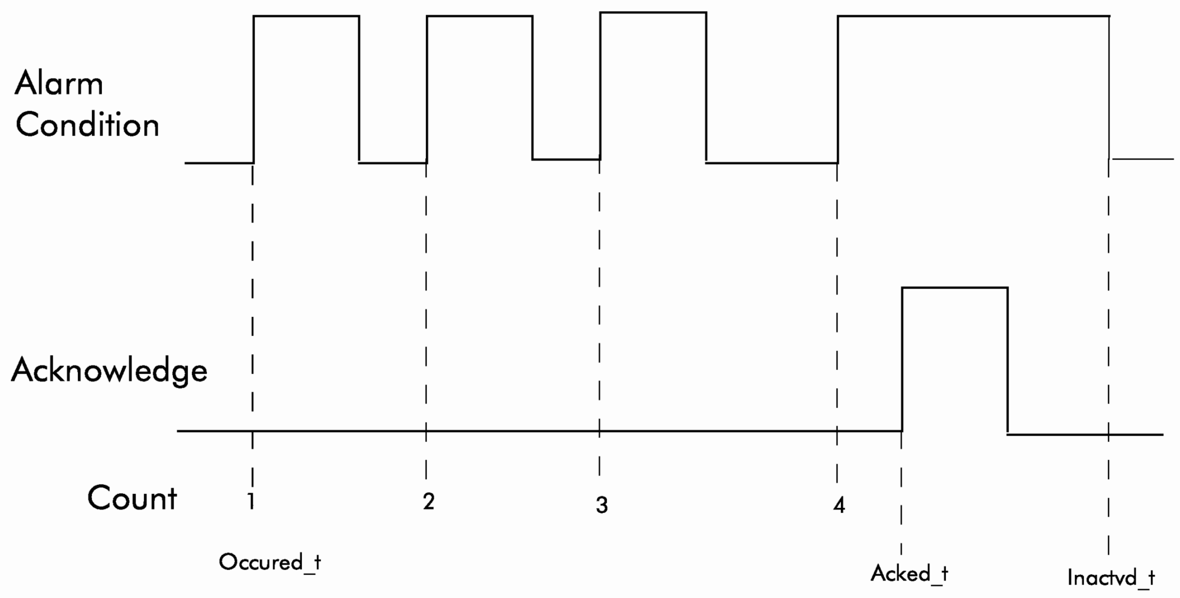
This parameter refers to the area which the alarm entry currently specified by Act_Index came from

Level (L)

The level of alarm at the position specified by Act_Index. Level is used to indicate the alarm priority. This allows different alarms to be prioritised and warning messages and corrective action to be controlled accordingly. The highest level alarm in the buffer is indicated by the parameter MXLevel.

Count (C)

The number of times the alarm at the position specified by Act_Index has become active. An alarm may be registered a number of times without the alarm being acknowledged (particularly during commissioning). Rather than creating a separate entry in the current alarm buffer for each occurrence, the first time the alarm became active is logged and the number of times it has subsequently changed state from inactive to active is counted.



AlmState(ALS)

This parameter indicates the state of the alarm at the position specified by Act_Index.

Value	**State**	**Meaning**
0	Empty	No alarm at this position. This is the state of an alarm that was acknowledged and is no longer active i.e. a dead alarm
1	NactNak	Not active, not acknowledged. An alarm which disappeared before it was acknowledged
2	ActAck	An alarm which is currently active but has been acknowledged
3	ActNak	An alarm which is currently active and has not yet been acknowledged

Table 20-3 States that may be displayed

Occurred_t (OCT)

Time that alarm currently specified by Act_Index became active.

Occured_p (OCP_

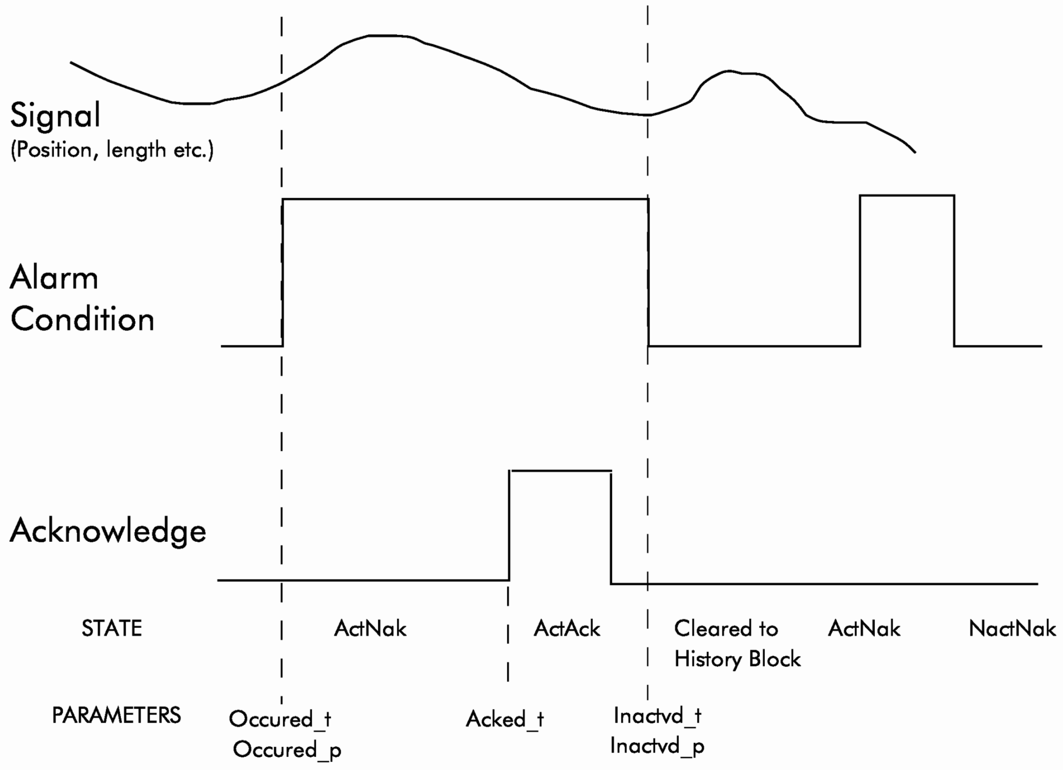
A value that indicates the position e.g. length at which the alarm specified byAct_Index became active.

Acked_t(AKT)

The time that alarm (Act_Index) was inactivated.

Inactvd_p (IAP)

A value that indicates the position e.g. length at which the alarm specified by Act_Index became inactive. The following diagram summarises the function of these parameters.



Alm_Str (STR)

The value of the alarm string for alarm specified by Act_Index. The alarm string will typically be used to store a text message associated with that alarm.

Ack_All (AA)

A boolean input which can be used Acknowledge all alarms in this buffer. Primarily intended for use during commissioning when multiple alarms may be active.

Clr_Nact (CCN)

A boolean input which will acknowledge (and therefore clear) all inactive alarms in this buffer.

General Usage

Examining alarms

When there are a number of alarms in the alarm buffer, they can be examined one at a time by setting the Index input. If this is set to an existent alarm number, all the relevant parameters will be displayed on the outputs of the blocks.

> Note that acknowledged time, inactivated position and inactivated time will be zero unless the appropriate action (acknowledgement or alarm off) have taken place.
>
> Note that alarm one is always the most recent alarm, unless the alarm handler is full (128 Alarms). If the alarm handler is full, a new alarm will not be logged unless its level is higher than the lowest level in the buffer.

For example, if alarm 20 in the buffer has level 3, all other alarms have level 4, and the alarm buffer is full. If another level 3 alarm occurs, it will not be logged. If a level 4 alarm occurs, alarm 20 will be cleared, and the new alarm added at position 1.

Acknowledging alarms

To acknowledge the currently displayed alarm, the Ack input is set. This input is cleared own by the block.

All alarms in the buffer can be acknowledged by setting the Ack_All parameter. This is cleared down by the block.

To remove all the inactive alarms from the buffer, set the Clr-Nact input, which is cleared by the block.

Detecting new alarms

A new alarm can be detected by monitoring the New_Alm input/output. This will be set by the block and clear by the user programme.

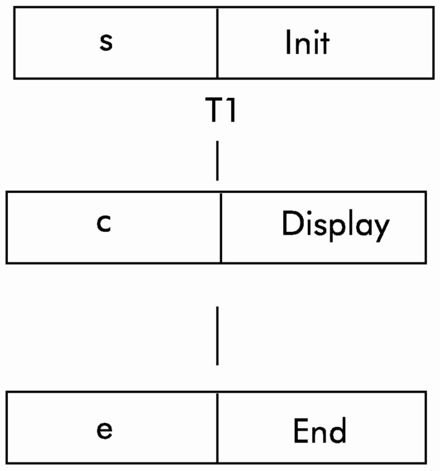
ST Fragment showing panel access to alarm block

```
PROGRAMME PANEL1 (*16 Nov-1992-14:43:41*)

VAR

      (*SYSTEM)
      PocsSTATE:PcsSTATE ;
      Tsk_10ms:Task     (Interval :=T#10ms
                         Priority  :=0);
      Tsk100ms:Task     (Priority :=1):
      Messages:Messages         ;
      RT-Clock:RT_Clocks;

      (*COMMS*)
      panel    :Euro_Panel       (Port    :='OA");
```

```
     (*USER_VAR*)
     level    :Boolean         ;
     alm      :Integer

(*SLAVE_VARS*'
   bool1      :Slave_Bool    (Address    :='Epbool1'):
   bool2      :Slave_Bool    (Address    :='EPbool2'):
   int1       :Slave_Int     (Address    :=E'Pint');
   int2       :Slave_Int     (Address    :='EPint2');
   time1      :Slave_Time    (Address    :='EPtime1');
   time2      :Slave_Time    (Address    :='EPtime2');
   strl       :Slave_Str     (Address    :='EPstrl');
   str2       :Slave_Str     (Address    :='EPstr2');

     (*STEPS *)
   MAIN       :Macro   ;
   PANEL      :Macro   ;
   Init       :Step            ;
   Display             :Step            ;
   End        :Step            ;

     (*ALARMS*)
buff    :Cur_Alms        (BufId     :1,
                         LogId     :=1,
                         CommsAddr :='EBr00);

high    :Sensor16        (Type:=1,
                         AreaStart :=1,
                         BufId     :=1,
                         Alm_Str   :+'Process Value Over Range Loop');

low     :Sensor16                  (Type         :=2,
                         AreaStart :=1,
                         BufId     :=1,
                         Alm_Str   :="Process Value Under Range Loop');

     (*Internal Variables*)
              (*SYSTEM*)

              (*COMMS*)

              (*USER_VAR*)

              (*SLAVE_VARS*)
```

```
        real1_Value      :REAL;
        real2_Value      :REAL;
        real3_Value      :REAL;
        int1_Value       :DINT;
        time1_Value      :TIME;
        time2_Value      :TIME;
        str1_Value       :STRING;
        str2_Value       :STRING;

        (*STEPS*)
        (*LOADS*)

        (*ALARMS*)
        buff_Index               :DINT;

END-VAR

(*function block instantiations*)
INITIAL_STEP EXECUTE_100)
buff    (Index                   :=buff_Index);
str2    (Value           :=str2_Value);
str1    (Value           :=str1_Value);
bool2   ();
bool1   ()
alm     ()
low     (AlmState1       :=load1.Main_Pv<-50,
        (AlmState2       :=load2Main_PV<-50,
        (AlmState3       :=load3.Main_PV<-50,
        (AlmState4       :=load4.Main_PV<-50,
high    (AlmState1       :=load1.Main_PV>30,
        AlmState2        :=load2.Main_PV>30,
        AlmState3        :=load3.Main_PV>30,
        AlmState4        :=load4.Main_PV>a50)
panel ():
RT_Clock ():
Messages ():
Tsk100ms ():
PcsSTATE ():
END_STEP

INITIAL_STEP EXECUTE_10
level ():
Tsk_10ms():
END_STEP
```

```
(*MACRO:ALARMS*)
STEP ALARMS:
```



```
*)

(*SINGLE SHOT*)
INITIAL STEP     ;
     (*Set  up the display and parse the page*)
     panel_Key_Pressed             :=22(*No_Key*):
     buff-Index                    :=1
     (Display the contents of str2 on the top line*)
     (*with enumerated output of alarm states*)
     panel _Format_A      :='str2:32C,int1(_,NactNak,Act
                                       Ack,ActNack)';
     (*Place "Acknowledge' above Fa, "Exit" above F3*)
     panel_Format_B       :="@0:1,"Acknowledge",@30,"Exit"';
     panel_Format_C       :=";
     panel_Change_Page    :=2(Nxt_pge*);
END-STEP
TRANSITION
     FROM INIT
     TO DISPLAY
:=1;(*NULL transition - default TRUE*)
END TRANSITION
(*CONTINUOUS*)
STEP Display
     (*Clockwise for next alarm*)
     IF panel.Key_Pressed=10(*Clkwise*) THEN
     buff_Index           :=buff.Index + 1;
     panel_Keyt_Pressed   :=22(*No_Key*);
     END_IF;
     (*Anticlockwise for previous alarm*)
     If panel,Key-Pressed= 11 (*AClkwse*) THEN
     buff-Index           :=buff.Index - 1;
     END_IF:
     (*F1 for acknowledge*)
     IF panel.Key_Pressed = 14 (*F1*) THEN
     buff_Ack             :=1 (*0n*);
     panel_Key_Pressed    :=22(*No_Key*);
     END_IF;
     (*Display "No Alarms" if there are no alarms!*)
     IF buff.NumAlarms - = 0 THEN
     str2_Value           :='No Alarms';
     int1_Value           :=0;
     ELSE
                  (*Limit Index to between 1 and buff.NumAlarms*)
                  IF buff.Index = 0 THEN
                  buff_Index        :=1;
     ELSIF buff.Index>buff.NumAlarms;
     buff_Index           :=buff.NumAlarms;
     END_IF;
     (*Display alarm string (set on sensor) and current state*)
     str2_Value           :=CONCAT(IN1:= BUFF.ALM_STR
     ,IN22:= DINT_TO_STRING(IN:=buff.Area)):
     intl-Value           :=buff.State;
     END-IF;
END-STEP

TRANSITION
     FROM Display
     TO End
:=
         panel.Key-Pressed = 16(F3*);
END_TRANSITION

(*SINGLE SHOT*)
STEP End:
END_STEP

END_STEP (*ALARMS*)

TRANSITION

     FROM ALARMS (*MACRO*)
     TO TOP
```

```
:=1;(*NULL transition - default TRUE*)
END_TRANSITION

END_STEP (*PANEL*)
END_STEP  (*MAIN*)
END_PROGRAM
```

Alarm Communications

The following provides further details on access to the contents of the current alarm buffer via the communications interfaces.

Eurotherm Bisync interface

The data is accessed by means of the slave parameter with the address set by 'CommsAdd'. The fields within the composite parameter described earlier are in the same order as the entries listed in the tables.

As an example, to read entry 3 in the alarm list, the 'Index' is set to 3:

EOT**0000**STX**A90**RS>**3**ETX?

The composite parameter may then be read back:

EOT**0000A90**ENQ

The reply is:

STX**A90**RS>**3**us>**type**us>**area**us.**state**us>**level**

us>**count**us>**otime**us@**oproduced**

us>**atime**us>**itime**us?@**iproduced**

In order to acknowledge alarm TYPE 1, AREA 4:

EOT**0000**STX**A90**RS US>**1**us>**4**ETX?

Packed Alarm Access (JBus/MODBUS or Eurotherm Bisync)

The packed alarm string appears as a consecutive set of registers under JBus, or as a string under Eurotherm Bisync. Each alarm entry comprises four bytes so that 32 alarms may be read using 128 byte access.

Each entry takes the form:

Active (255 for active, 0 for inactive)

Type

Acknowledge (128 for not acknowledged, 0 for acknowledged)

Area

The alarm type is used as an index into an array of strings. For instance, type 1 may be an over temperature alarm, type a low alarm, type 3 deviation etc. the area is then appended to this string to form the overall alarm message, e.g. type 2 area 3 could be read as 'low alarm zone 3'.

Alarms are acknowledged by writing to the first two bytes of the string or if JBus/MODBUS communications are used, the first register. The first byte is the type, the second is the area.

Parameter Attributes

Name	Type	Cold Start	Read Access	Write Access	Type Specific Information	
BufId	DINT	1	Oper	Oper	High Limit Low Limit	255 1
LogId	DINT	1	Oper	Oper	High Limit Low Limit	255 1
CommsAddr	STRING	"	Oper	Oper	Max 13 Characters	2
Index	DINT	0	Oper	Oper	High Limit Low Limit	255 1
Ack	BOOL	Off (0)	Oper	Oper	Senses	Off (0) On (1)
Pack_Addr	STRING	"	Oper	Oper	Max 13 Characters	
New_Alm	BOOL	No (0)	Oper	Oper	Senses	No (0) New_Alm (1)
Ack_All	BOOL	Off (0)	Oper	Oper	Senses	Off (0) Ack_All(1)
Clr_Nact	BOOL	Off (0)	Oper	Oper	Senses	Off (0) Clr_Nact (1)
Status	BOOL	NOGO (0)	Oper	Oper	Senses	NoGo (0) Go(1)
Error_No	DINT	0	Oper	Oper	High Limit Low Limit	255 0
LiveAlarms	DINT	0	Oper	Oper	High Limit Low Limit	255 0
NumAlarms	DINT	0	Oper	Oper	High Limit Low Limit	255 0
MxLevel	DINT	0	Oper	Oper	High Limit Low Limit	255 0
ActIndex	DINT	0	Oper	Oper	High Limit Low Limit	255 0
Type	DINT	0	Oper	Oper	High Limit Low Limit	255 0

Table 20-4 Cur_Alms Parameter Attributes

Name	**Type**	**Cold Start**	**Read Access**	**Write Access**	**Type Specific Information**	
Area	DINT	0	Oper	Oper	High Limit Low Limit	255 0
Level	DINT	0	Oper	Oper	High Limit Low Limit	255 0
Couint	DINT	0	Oper	Oper	High Limit Low Limit	255 0
AlmState	ENUM	Empty (0)	Oper	Oper	Senses	Empty (0) NactNak (1) ActAck (2) ActNak (3)
Occurred_t	DATE_AND_TIME		Oper	Oper	High Limit Low Limit	On (1)
Occurred_p	REAL	0	Oper	Oper	High Limit Low Limit	On (1)
Acked_t	DATE_AND_TIME		Oper	Oper	High Limit Low Limit	On (1)
Inactvtd_t	DATE_AND_TIME		Oper	Oper	High Limit Low Limit	On (1)
Inactvtd_t	DATE_AND_TIME		Oper	Oper	High Limit Low Limit	
Alm_Str	STRING	0	Oper	Oper	Max length	255 Characters

Table 20-4 Cur_Alms Parameter Attributes (continued)

HISTORY FUNCTION BLOCK

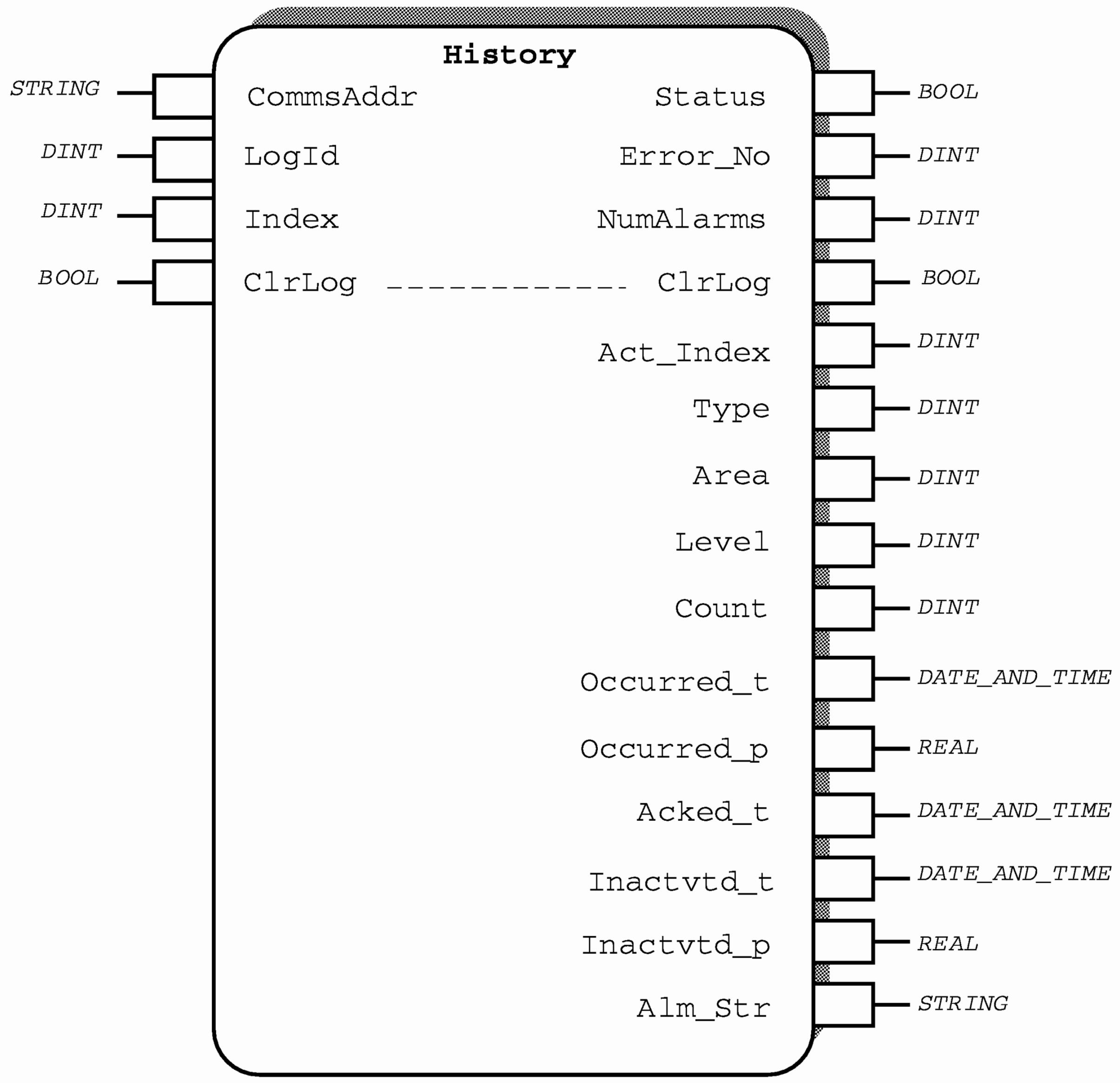


Figure 20-3 History Function Block Diagram

Functional Description

The History function block contains an array of the last 128 cleared alarms, and allows access to the elements of this array by means of an index input. It is intended for use with the Cur_Alms function blocks to provide a system for recording historical alarms and user defined events within the PC3000 controller.

The block also contains on internal communications slave variable. This provides a Eurotherm Bisync interface and allows access to a composite or multi-element parameter containing all alarm information. Access via MODBUS/JBUS is NOT supported on this block (c.f. the Current Alarm buffer).

The historical alarm buffer is associated with one or more Cur_Alms function blocks (function blocks which store data associated with live or current alarm conditions) by means of a log identifier. This allows the use of more than one history alarm buffer in the PC3000. It also allows multiple current alarm buffers to store historical alarms in the same historical log. In this case all current alarm buffers would be set with the same log identifier.

Function Block Attributes

Type:	6671
Class:	ALARMS
Default Task:	Task_1
Short List:	LogID, Index, NumLogged, Status
Memory Requirements:	5348 Bytes

Parameter Descriptions

LogID (LID)

This is the identifier (ID) for the alarm log. It is used to connect or associate a number of alarm handlers to an alarm log. Multiple alarm handlers can log their alarms in a single alarm log by having the same LogID. The LogID value must lie in the range 0 to 9.

Index (I)

This is an index into the array of logged alarms. It may take any value in the range 0 to 127. However, the alarm log will only contain valid historic alarm data in the range 0 to NumLogged and the value of the parameter Index should be restricted to this range.

CommsAddr (ADD)

This is the comms address for a multi-element or composite slave parameter. This address provides access via the Eurotherm Bisync protocol to the information contained in the alarm array. The data may be accessed as a single multi-element parameter, or as a number of separate elements. Reading the data as the slave address will return the multi-element parameter. Access to subsequent addresses will return only one element. It is not possible to specify data format for the individual element and the default format is assumed.

Offset	Parameter	Type
0	Composite parameter	
1	Index	Integer
2	Type	Integer
3	Area	Integer
4	State	Integer
5	Level	Integer
6	Count	Integer
7	Occured Time	Integer
8	Occured Position	Real
9	Acked Time	Integer
10	Inactivated Time	Integer
11	Inactivated Position	Real

The offset refers to the offset from the base address. As an example, if the slave address is set up with an address of 'EBA90' and the associated Bisync slave block has a GID setting of '0' the following address would apply:

Offset	Parameter	Type
00 A90	Composite parameter	
00 A91	Index	Integer
00 A92	Type	Integer
00 A93	Area	Integer
00 A94	State	Integer
00 A95	Level	Integer
00 A96	Count	Integer
00 A97	Occured Time	Integer
00 A98	Occured Position	Real
00A99	Acked Time	Integer
00 A9	Inactivated Time	Integer
00A9	Inactivated Position	Real

For further details on alarm addressing see the section covering alarm communications.

ClrLog (CLR)

This is an input/output used to clear down the alarm log. It is set by the user and cleared by the block.

Status (ST)

This refers to the function block status and may take the states Go in normal operation or NOGO in the event of an error. The error is indicated by the parameter Error_No.

Error_No (ERR)

This indicates the function block error number if an error is detected when the PC3000 programme runs, the error will be indicated as shown in the table below and the block status will become NOGO

The following errors may be detected and indicated:

Error Number	**Error**	**Cause and Action**
105	Too many buffers	There are more than 20 buffers (Cur_Alarms and History function blocks). Any buffer indicating this error will not be operational and the number of buffers must be reduced.
106	Buffer ID out of range	The Log ID is less than 0 or greater than 9. Change the Buffer ID and sensor ID's associated with this buffer.
107	Duplicate ID	A buffer already exists with this ID. Change the buffer ID and sensor ID's associated with this buffer.

NumLogged (NL)

Indicates the total number of alarms in the buffer.

Act_Index (AIX)

This will be equal to Index if the parameter Index lies in the following range"

0<Index<=NumLogged

It will be equal to zero for values of Index outside this range.

Type (TYP)

This indicates the alarm type at the position in the log specified by Act_Index.

Area (A)

This parameter refers to the area which the alarm entry currently specified by Ac_Index came from.

Count (C)

The number of times the alarm currently specified by Act-Index became active.

Occurred_t(OCT)

Time that alarm currently specified by Act_Index became active.

Occurred_p (OCP)

A value that indicates the position e.g. length at which the alarm specified by Act_Index became active.

Acked_t(AKT)

The time that alarm specified by Act_Index became acknowledged.

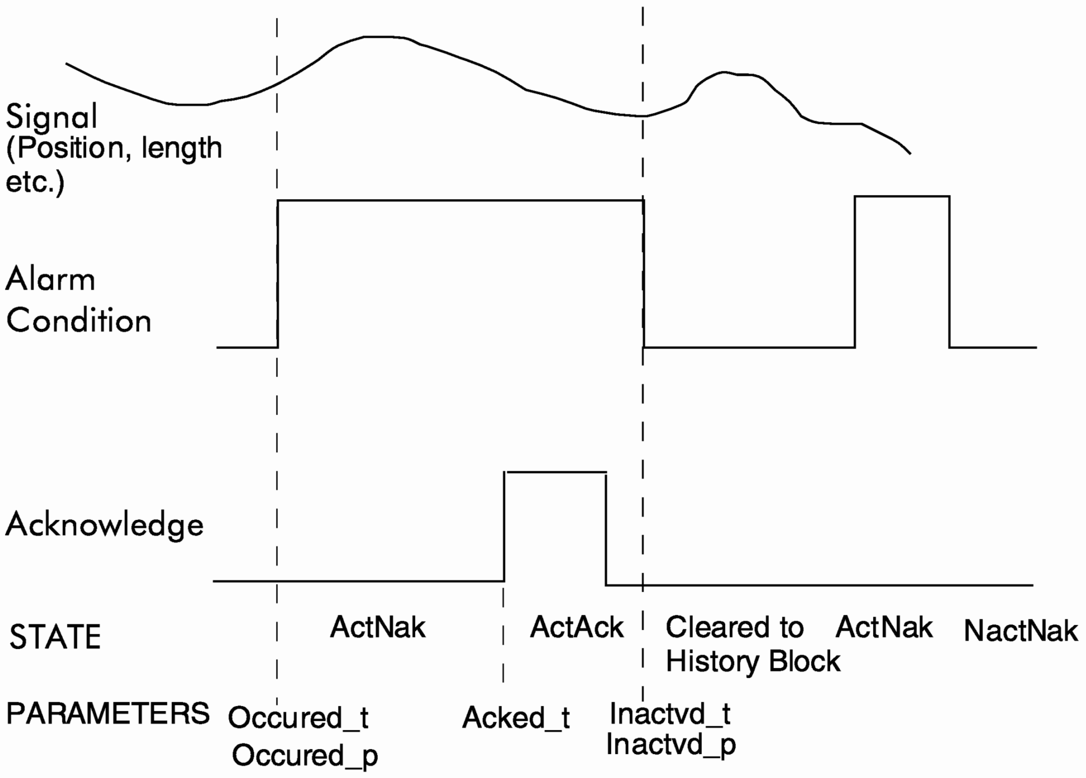
Inactvtd_t(IAT)

The time that alarm specified by Act_Index was inactivated.

Inactvtd_P (IAP)

A value that indicates the position e.g. length at which the alarm specified by Act_Index became active.

The following diagram summarises the function of these parameters:



Alm_Str (STR)

Current value of alarm string for alarm specified by Act_Index. The alarm string will typically be used to store a text message associated with that alarm.

Alarm Communications

The following provides further details or access to the contents of the historical alarm buffer via the communications interfaces.

Eurotherm Bisync interface

The data is accessed by means of the slave parameter with the address set by 'Comms Addr'. The fields within the composite parameter described earlier are in the same order as the entries listed in the tables.

As an example, to read entry 3 in the alarm list, the 'Index" is set to 3:

EOT**0000**STX**A90**RS>**3**ETX?

The composite parameter may then be read back:

EOT**0000A90**ENQ

The reply is:

STX**A90**RS>**3**us>**typ**eus>**area**us.**state**us>**level**

us>**count**us>ou**time**us@o**produced**

us>**atime**us>**itime**us>@**iproduced**

Parameter Attributes

Name	Type	Cold Start	Read Access	Write Access	Type Specific Information	
CommsAddr	STRING	''	Oper	Oper	Max 13 Characters	
LogId	DINT	1	Oper	Oper	High Limit Limit	255 Low 1
Index	DINT	0	Oper	Oper	High Limit Limit	255 Low 1
ClcLog	BOOL	Off (0)	Oper	Oper	Senses	Off (0) On (1)
Status	BOOL	NOGO (0)	Oper	Oper	Senses	NoGo (0) Go(1)
Error_No	DINT	0	Oper	Oper	High Limit Low Limit	255 0
NumLogged	DINT	0	Oper	Oper	High Limit Low Limit	255 0
ActIndex	DINT	0	Oper	Oper	High Limit Low Limit	255 0
Type	DINT	0	Oper	Oper	High Limit Low Limit	255 0
Area	DINT	0	Oper	Oper	High Limit Low Limit	255 0
Couint	DINT	0	Oper	Oper	High Limit Low Limit	255 0
Occurred_t	DATE_AND_TIME		Oper	Oper	High Limit Low Limit	
Occurred_p	REAL	0	Oper	Oper	High Limit Low Limit	
Acked_t	DATE_AND_TIME		Oper	Oper	High Limit Low Limit	
Inactvtd_t	DATE_AND_TIME		Oper	Oper	High Limit Low Limit	
Inactvtd_t	DATE_AND_TIME		Oper	Oper	High Limit Low Limit	
Alm_Str	STRING		Oper	Oper	Max length	255 Characters

Table 20-5 History Parameter Attributes

SENSOR FUNCTION BLOCKS

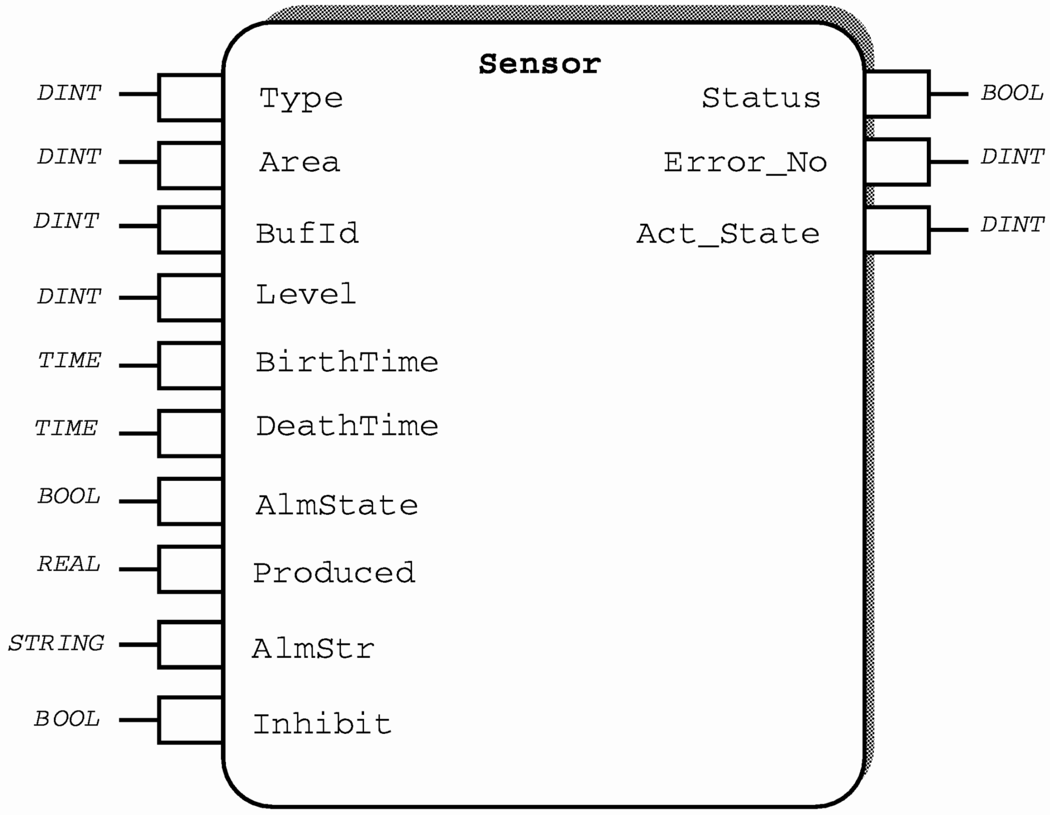


Figure 20-4 Sensor Function Block Diagram

Functional Description

Alarm sensors are the information gathering mechanism in the alarm subsystem. Alarm sensors gather information such as the time at which the alarm occurred, and the source of the alarm condition. This date is then logged or stored in an alarm buffer block Cur_Alms. This live or current alarm buffer is associated with the alarm sensors by means of a buffer identifier parameter.

Different sensor may be associated with separate alarm buffers if necessary by assigning different buffet identifier values on each group of sensors.

Sensors may be used to detect an user defined alarm condition or event or they may be used in conjunction with a Detector block in order to provide information relating to 'standard' alarm strategies. "Standard" alarms include High, low, deviation etc. A user defined alarm might require that a value is exceeded and that a particular time has elapsed. In the case of user defined alarms, the condition is defined by any ST expression which evaluates to TRUE or FALSE. this expression may be soft wired to the alarm sensors alarm state input.

When the alarm state changes the state is passed to the current alarm buffer.

Function Block Attributes

Type:	662
Class:	ALARMS
Default Task:	Task_1
Short List:	Type, Area, BufID, Alm_Str
Memory Requirements	326 Bytes

Parameter Descriptions

The Sensor function block provides the following parameters:

BufId (BID)

This alarm buffers identifier (ID). It is the reference by which the alarm sensor is are connected to or associated with the Cur_Alms function block.

Type and Area (TYP) and (AR)

The type and area can be used in two different ways. In some alarm systems alarms are grouped into a type (e.g. 'process value over range') and an area (e.g. Loop a). Each alarm sensor's type and area combined give a unique reference (there will only be one sensor with type 1 and area 5, for instance). This system of alarms is particularly useful with Xycom terminals, where a complete alarm message (e.g. 'Process Value Overrange Loop 1') can be generated from two bytes of data.

Alarm systems with no concept of type and area just need a unique number to identify each alarm sensor block.

Level (L)

Different alarms may require different actions in the control system. For instance, a deviation from setpoint alarm may just be a warning, whereas process value over range may require that the control system, shuts down. To allow this, each alarm sensor has a Level input. The alarm log records the level and the maximum level in the alarm log.

AlmState (S)

This is a boolean parameter which is wired with the fault condition. For instance, an over temperature alarm may have

```
sensor.1.State := AI1.Process_Val > alm_Level.Value:
```

Produced (P)

The produced parameter is used in processes where a position or volume of product must be logged in the alarm buffer. For instance, a spark fault in a cable needs the position not the time logged.

Alm_Str (STR)

Applications using simple panels such as a VT100 terminal or the Euro Panel cannot calculate an alarm string from a numeric reference. Each sensor function block therefore has an alarm string input for messages.

Inhibit (INH)

The inhibit function forces the alarm to the inactive state. It is used to disable alarms during certain phases of the process. For instance, deviation alarms may be turned off while an over door is opened. If the almState is active when inhibit is set low, a new alarm will be generated immediately.

Status (ST)

The status is a simple Go/NOGO parameter. The reason for the error is indicated by the Error_No parameter.

Error_No (ERR)

This indicates the function block error number. If an error is detected when the PC3000 programme runs, the error will be indicated as shown in the table below and the block status will become NOGO>

The following errors may be detected and indicated:

Error Number	Error	Cause and Action
103	No current alarm buffer	There is no Cur_Alarms block with a buffer identifier of Bufld in the programme. Check the Bufld parameters on the sensor and current alarm blocks.
133	No more space in the log	The current alarm block with a buffer identifier of Bufld has 128 uncleared alarms in it. The alarm which has just occurred does not have a higher level than the lowest level alarm in the log and has not been added. There are too many alarms occurring in a short period. If this occurring during start up or commissioning, disable alarms using the Inhibit input on the sensors.

Time Hysteresis ((Birth Time, (BT) and Death Time, (DT))

Time Hysteresis requires that the alarm remains in a particular state for a time before the alarm is actioned. The sensor has an Act_State output, so that the actual alarm state after hysteresis can be monitored.

The Inhibit or function disables the alarm regardless of time hysteresis. When Inhibit is turned off, however, the alarm must remain active for the Birth/Time before an alarm is generated.

Name	Type	Cold Start	Read Access	Write Access	Type Specific Information	
Type	DINT	0	Oper	Oper	High Limit Low Limit	255 0
Area	DINT	0	Oper	Oper	High Limit Low Limit	255 0
BufID	DINT	1	Oper	Oper	High Limit Low Limit	255 0
Level	DINT	1	Oper	Oper	High Limit Low Limit	255 1
Birth Time	TIME	0ms	Oper	Oper	High Limit Low Limit	
Death Time	TIME	0ms	Oper	Oper	High Limit Low Limit	
AlmState	BOOL	Off (0)	Oper	Oper	Senses	Off (0) On (1)
Produced	REAL	0	OPER	BLOCK	High Limit Low Limit	
Alm_Str	STRING	0	Oper	Oper	Max length	255 Characters
Inibit	BOOL	Off (0)	Oper	Oper	Senses	Off (0) On (1)
Status	BOOL	NOGO (0)	Oper	Oper	Senses	Off (0) On (1)
Error_No	DINT	0	Oper	Oper	High Limit Low Limit	255 0
Ac_State	BOOL	Off (0)	Oper	Oper	Senses	Off (0) On (1)

Table 20-6 Sensor Parameter Attributes

SENSOR 16 FUNCTION BLOCK

To reduce the time spent instantiating function blocks, there is also an alarm sensor 16 block, which contains 16 nested sensor function blocks. The blocks have a common Type, and contiguous Area parameters. They also have common hysteresis values, alarm string, and inhibit inputs.

DETECTOR FUNCTION BLOCK

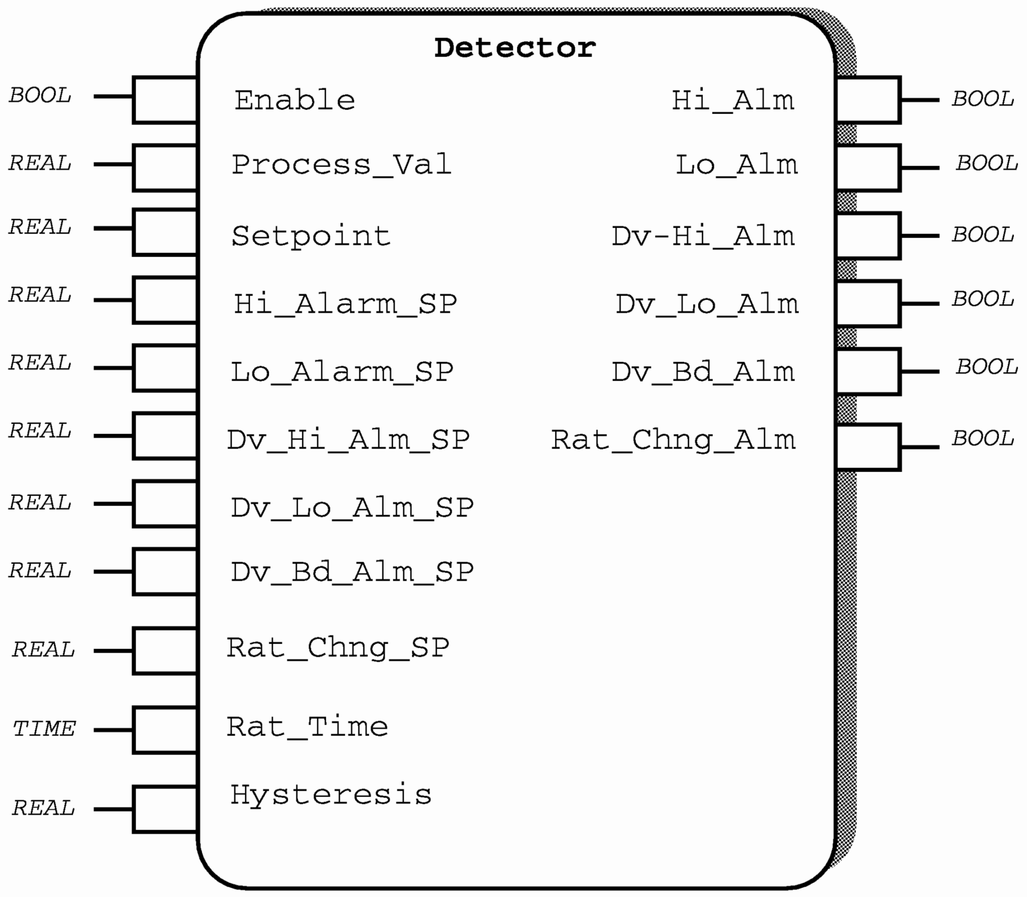


Figure 20-5 Detector Function Block Diagram

Functional Description

The Detector function block takes as its input the process value and set point together with a set of limits within which the process value should operate. The block then has boolean outputs corresponding to each of these limits. The limit checks performed are

1. High limit
2. Low limit
3. High deviation limit
4. Low deviation limit
5. Deviation band limit
6. Rate of change limit

A hysteresis function is also built into all the limit checks except the rate of change and an inhibit is provided to turn all outputs off.

The function block is used in conjunction with an alarm sensor function block which stores data such as time at which alarm occurred, etc. The relevant detector block output should be wired to an alarm sensor AlmState input.

Function Block Attributes

Type:	6674
Class:	ALARMS
Default Task:	Task_1
Short List:	Enable, Process_Val, Setpoint, Dv_Bd_Alm
Memory Requirements:	66 Bytes

Parameter Descriptions

The Detector function block provides the following parameters:

Enable (EN)

When the Enable input is off (*0*) then all outputs are Off (*0*) and the averaging for the rate of change alarm is re-set. When Enable is on all outputs act as described below.

Process_Val(PV)

The process value to be monitored.

Setpoint(SP)

The corresponding setpoint for the process, used for deviation alarms.

Hi_Alarm_SP (HAS)

The level above which the high alarm will become active.

Lo_Alarm_SP (LAS)

The level below which the low alarm will become active.

Dv_Hi_Alm_SP (DHS)

The deviation from setpoint above which the deviation high alarm will become active.

Dv_Lo_Alm_SP (DLS)

The deviation from setpoint below which the deviation low alarm will become active.

Dv_Bd_Alm_SP)DBS)

The deviation from setpoint beyond which the deviation band alarm will become active.

Rat_Chang_SP (RCS)

The change in process value over Rat_Time beyond which a rate of change alarm will be produced.

Rat_Time (RT)

The time over which the rate of change alarm is calculated.

Hysteresis (HYS)

The hysteresis level for the high, low, high deviation, low deviation and deviation band alarms.

The function of each of the outputs is described below:

Hi-Alm (HA)

If Process_Val<Hi_Alm_SP - Hystersis:	Hi_Alm=Off(*0*)
Hi-Alm_SP - Hysteresis<Process_Val<Alm_SP	Hi_Alm=unchanged
Hi_SAlm-SP<Process_Val:	Hi_Alm=On(*0*)

Lo-Alm (LA)

If Process_Val>Lo-Alm_SP+Hysteresis: Lo_Alm=Off(*0*)

Lo_Alm_SP+Hysteresis>Process_Val>Lo_Alm_SP: Lo_Alm=unchanged

Lo_Alm_SP>Process_Val: Lo_Alm+On(*1*)

Dv_Hi_Alm (DHA)

If deviation<Dv_Hi_Alm_SP-Hysteresis: Dv_Hi_Alm=Off(*0*)

Dv_Hi_Alm_SP-Hysteresis_deviation,Dv_Hi_Alm-SP:

Dv_Hi_Alm=unchanged

Dv_Hi_Alm_SP<deviation Dv_Hi_Alm=On(*1*)

Deviation = Process_Val - Setpoint

Dv_Lo_Alm (DLA)

If deviaton,Dv_Lo-Alm_SP - Hysteresis: Dv_Lo_Alm=Off(*0*)

Dv_Lo_Alm_SP - Hysteresis<deviation<Dv_Lo_Alm-Sp:

Dv_Lo_Alm_=unchanged

Dv_Lo_Alm_SP<deviation: Dv_Lo_Alm=On(*1*)

Deviation + Setpoint - Process_Val

Dv_Bd-Alm ((DBA)

If deviation<Dv_Bd_Alm_SP-Hysteresis: Dv_Bd_Alm=Off(*0*)

Dv_Bd_Alm_Sp-Hysteresis<deviation<Dv_Bd_Alm_SDv_Bd_AlmS_P

Dv_Bd_Alm=unchanged

Dv_Bd_Alm-SP<deviation: Dv_Bd_Alm=On(*1*)

Deviation + Setpoint - Process_Val

Rat_Chng_Alm (RCA)

The process value is filtered by taking the previous filtered value and the current process value:

$PV_{filtered}(n)=(3*PV_{filtered}(n-1)+PV(n)4$

Note that this algorithms does not take into account startup conditions and these should be considered in the user programme.

The rate limit check will be performed on the filtered Process_Val by taking the absolute difference PVfiltered(n)-PVfiltered(n-1) and dividing by the sample time to provide a rate. This will then be compared with the rate given by the Rat_Chang_SP/Rat_Time

To summarise:

Rat_Chng_Alm= $\left| PVfiltered(n)-PVfiltered(n-1) \right| >(Rat\_Chang\_SP/Rat\_Time)$

Name	Type	Cold Start	Read Access	Write Access	Type Specific Information	
Enable	BOOL	Off (0)	Oper	Oper	Senses	Off (0) On (1)
Process_Val	REAL	0	Oper	Oper	High Limit Low Limit	
Setpoint	REAL	0	Oper	Oper	High Limit Low Limit	
Hi_Alm_SP	REAL	0	Oper	Oper	High Limit Low Limit	
Lo_Alm_SP	REAL	0	Oper	Oper	High Limit Low Limit	
Dv_Hi_Alm_SP	REAL	0	Oper	Oper	High Limit Low Limit	
Dv_Lo_Alm_SP	REAL	0	Oper	Oper	High Limit Low Limit	
Dv_Bd_Alm_SP	REAL	0	Oper	Oper	High Limit Low Limit	
Rat_Chang_SP	REAL	1	Oper	Oper	High Limit Low Limit	
Rat_Time	TIME	1m	Oper	Oper	High Limit Low Limit	
Hi_Alm	BOOL	Off (0)	Oper	Oper	Senses	Off (0) On (1)
Lo_Alm	BOOL	Off (0)	Oper	Oper	Senses	Off (0) On (1)
Dv_Hi_Alm	BOOL	Off (0)	Oper	Oper	Senses	Off (0) On (1)
Dv_Lo_Alm	BOOL	Off (0)	Oper	Oper	Senses	Off (0) On (1)
Dv_Bd_Alm	BOOL	Off (0)	Oper	Oper	Senses	Off (0) On (1)
Rat_Chng_Alm	BOOL	Off (0)	Oper	Oper	Senses	Off (0) On (1)

Table 20-7 Detector Parameter Attributes