CHAPTER 23  Alarm & Event Programming

Objectives

After completing this chapter, you should have the knowledge to:

- Understand alarm programming.
- Apply the various techniques to capture alarm and event information.

Many PLC applications require some form of alarming when abnormal conditions or events occur. PLCs are well suited for this task and in fact, many PLC manufacturers provide special instructions designed just for this purpose. Some of these special instructions will be covered later in this chapter.

Alerting operating or maintenance personnel of abnormal conditions or problems saves time and increases machine or process up-time. When detailed alarming is provided, many problems can be resolved by operators themselves rather than calling a technician to find the problem saving time and increasing productivity.

When a PLC system is connected to one of the many HMI interfaces available on the market today, the HMI can provide all of the alarming functions necessary with minimal additional PLC programming. Most HMI software packages provide some form of alarming and event recording. These HMI systems continually monitor the PLC’s internal data files or tags and based on how the HMI is programmed will alarm accordingly. When HMI interfaces with built-in alarming features are not part of the overall system or when events must be captured faster than an HMI system can monitor, the alarming and event recording must be programmed in the PLC. This chapter is dedicated to showing the reader some of the techniques and instructions used to program alarm and event information in PLCs.

BASIC ALARM LOGIC

Alarms and events can be triggered from digital information such as a motor failing to start or run. They can be triggered from process analog values such as a high temperature or low pressure. Whether monitoring a digital or analog value the end result is the same, a bit or tag is typically set to a 1 or ON indicating that a problem has been detected. Figure 23-1 illustrates an example of this basic logic.
Rung 1 in Figure 23-1 monitors the motor on/off switch and corresponding motor starter status input to set an alarm if the motor is not running when the switch is in the ON position. The second rung monitors an analog temperature input (N7:1) and alarms if the temperature exceeds 100°F. In both rungs the output is an internal bit or tag that, when ON, indicates an abnormal condition. This internal bit or tag can be replaced with an actual output that illuminates a light, sounds a horn, or left as an internal memory address that is monitored by an HMI interface system.

After studying Rung 1 in Figure 23-1, you should see a basic problem with this alarm logic. When the motor switch is turned ON, the status input from the motor starter does not change state immediately and therefore the motor alarm will be ON for a fraction of a second as the motor contactor changes state. This fraction of a second may not seem like much but it can trigger false alarms that can cause problems with other PLC logic that uses this alarm information or falsely turning on an alarm output.

A similar problem can be seen in the second rung with the over temperature alarm. If the temperature reached 101°F for only a split second causing the high temperature alarm to turn ON shutting down the process, the operator may not know what happened.

Both problems described above can be corrected with some simple changes to the alarm logic as shown below in Figure 23-2.
In the first run an On-Delay Timer (TON) has been added that allows for a small delay before the motor alarm is activated which is enough time for the motor contactor to change state and prevent a false alarm. The second run uses a sealing circuit to seal-in the alarm until the alarm is reset by the operator. In our example we have used the start push button as the reset method. Of course, the temperature must be below the alarm set-point before the alarm can be reset. The start button can be replaced with almost any desired action. See Chapter 22 for an example of a motor fault and monitoring logic that can be used to alarm abnormal motor conditions.

**CAPTURING ALARM EVENTS**

Capturing and retaining alarm information can be done in multiple ways. A simple method is to use a sealing circuit as was shown in Figure 23-2, Rung 2 and some means to rest the alarm. This method is non-retentive, meaning the alarm status will be lost if the processor loses power. Latching instructions can be used to make the alarm retentive as shown in Figure 23-3. Other methods, such as moving data, will be discussed later in this chapter.

![Figure 23-2 Enhanced Basic Alarm Logic](image-url)
You will notice in Figure 23-3 that the alarm reset (unlatch) rung is located above the alarm set (latch) rung. This is done to ensure that the alarm remains set for the entire PLC program scan if the start button is pushed while the alarm is still active.

**ANALOG RATE-OF-CHANGE ALARM LOGIC**

Some applications require that analog values such as temperature, pressure, level, etc. be monitored for a rapid rate of change and alarm if such a condition exists. The logic in Figure 23-4 uses a timer, compare, move, and bit instructions to create a rate of change alarm. Rung 2 in Figure 23-4 is a free running timer set to the evaluation period (30 seconds). This evaluation period can be set to any desired value by simply changing the timer preset value. Rung 4 compares the current analog value stored in F8:0 with the previous analog value (F8:1) that was stored at the end of the last evaluation period. The maximum allowable rate of change is stored in F8:2 and can be changed to any desired value. The alarm output is latched using the latch instruction and requires a corresponding unlatch instruction to reset the alarm. Rung 1 is used to store the current analog value when the PLC is first started so that a false alarm is not generated the first time the timer is done. This logic monitors for a positive (increasing) rate of change, which can be easily changed to alarm on a decreasing rate of change or both if desired.
Figure 23-4 Rate-of-Change Alarm Logic
ANNUNCIATOR LOGIC

Early alarming methods used an annunciator board to notify operators of problems. These annunciator boards used a light behind an opaque window. The opaque window would have the name of the alarm engraved on the window. When the alarm was active the window would be illuminated. These alarm annunciators would use three modes to indicate an alarm condition. If the alarm had been triggered but not yet acknowledged by the operator, the alarm window would flash indicating the problem. Once the alarm was acknowledged, the alarm window would stop flashing and either remain illuminated or turn off depending on the state of the alarm. Operators could monitor the annunciator board and tell which alarms had occurred (flashing), which are still active (illuminated), and which were not active (non-illuminated).

The PLC ladder logic to mimic the early annunciators is shown Figure 23-5. Here we have used the same over temperature alarm as was shown in Figure 23-2 and added the necessary logic so that the alarm functions as described above. The “Flasher” contact is a ½ second On/Off internal memory address. See Chapter 22 for an example on how to program this flasher.

![Figure 23-5 Basic Annunciator Type Logic](image)

The over temperature alarm could be a hardwired output to a light on an annunciators board or it could be an internal memory address monitored by an HMI that has been programmed to mimic an annunciator board.
If you only have a handful of alarms to program, then the method shown in Figure 23-5 will work just fine. But if you have more than a handful or even a few hundred alarms to deal with, then this method may become cumbersome requiring additional programming time, memory usage, and scan time. A much better way to handle a large number of alarms is to use bitwise logic along with word and file move instructions. If you recall from Chapter 6 “Digital Logic Gates” that digital logic notations such as AND, OR, XOR, etc. operate at the bit level to make logical decisions. If you are not familiar with digital logic notations then a review of Chapter 6 would be strongly recommended before proceeding.

In the example that follows we will program the alarm logic for 16 alarms (one 16 bit word) by manipulating words of data. Later in this chapter we will show how you can take this basic approach and deal with hundreds of alarms at one time. We will use the Allen-Bradley PLC-5 for this example, but any PLC with word, file, and bitwise operators can be used.

The first step is to create four (4) integer memory words (16 bit) and label as follows:

- Word 1 (N7:0) Raw Alarms – this word contains the actual alarm bits set in your user program.
- Word 2 (N7:1) Current Alarms – contains all current alarms that are either acknowledged or unacknowledged.
- Word 3 (N7:2) Acknowledged Alarms – contains the status of all acknowledged alarms.
- Word 4 (N7:3) Unacknowledged Alarms – contains the status of all unacknowledged alarms that are still active.

The second step is to program each of your raw alarm bits using one of the 16 bits available in word N7:0 as shown in Figure 23-6.
The third step is to program the logic to find all unacknowledged and acknowledged alarms. In Figure 23-7 we are using two Compute (CPT) instructions to perform this task. You will notice in Figure 23-7 that we are using logical operators in the expressions of each compute instruction to update the acknowledged and unacknowledged alarm words.

![Figure 23-7 Logic to Find Unacknowledged and Acknowledged Alarms](image)

To better understand what each expression is doing in the two CPT instructions refer to the matrix in Figure 23-8a (unacknowledged) and Figure 23-8b for acknowledged alarms. Only the first five bits of each word are shown for easy of understanding.

![Figure 23-8a Unacknowledged Alarm Matrix](image)
In step four we program the logic to clear the unacknowledged alarms and move the current alarms (N7:1) into the acknowledged alarm word when the acknowledged push button is pressed. See Figure 23-9.
Step five uses another CPT instruction to update the current alarm word (N7:1). See Figure 23-10.

Refer to the matrix in Figure 23-11 to better understand the logical operation being performed in the last CPT instruction of Figure 23-10.
Rungs 3, 4 and 5 in Figure 23-10 monitor all 16 alarms and provide the user with status information on each alarm such as whether the alarm has been acknowledged, unacknowledged, or currently in an active alarm state. This information can be used in many different ways such as turning on warning lights or annunciator boards, or sounding a horn if there are any unacknowledged alarms. The real benefit to this type of alarm logic is that it operates in a single scan and will trap the status of any alarm even if the alarm was only on for one scan.

**MULTIPLE ALARM & EVENT CAPTURING LOGIC**

To increase the number of alarms (above sixteen) is a simple matter of replacing the CPT instructions with file type instructions that operate on multiple words at one time such as the Allen-Bradley FAL instruction. The program in Figure 23-12 replaces the CPT, MOV, and CLR instructions with FAL, COP, and FLL instructions that monitor the status of 80 (16x5) alarms. You can increase or decrease the number of alarms by simply changing the length or words to operate on. It is recommended that if you have a large number of alarms or plan to increase the number in the future that you create a separate file for each alarm status (raw, current, unacknowledged and acknowledged) as shown in Figure 23-12.

N7:0 through N7:4 contain the raw alarms that are programmed into the user PLC logic. N10:0 through N10:4 contain the status of all current alarms, both acknowledged and unacknowledged. N11:0 through N11:4 contains all acknowledged alarms and N12:0 through N12:4 the unacknowledged alarms.
Figure 23-12 Alarm Logic Example For Up To 80 Alarms
The reset output instruction that follows each FAL instruction ensures that the FAL is reset before the next program scan so that no alarms are lost.

The examples shown so far in this chapter are only some of the many ways that alarming information can be handled in the PLC. Most HMI systems today have very extensive alarming and event recording capabilities that make programming alarm logic in the PLC almost obsolete. The questions you must ask yourself is, does the HMI update fast enough to capture your alarms without adding additional PLC logic.

**ALLEN-BRADLEY LOGIX5000 ALMD & ALMA INSTRUCTIONS**

The Allen-Bradley Logix family of PLC processors has two alarm/event instructions designed specifically for monitoring digital and analog data for abnormal conditions as programmed by the user. These Logix-based alarm instructions are available for use in relay ladder, structured text, and function block diagram programs. The advantage to using these two instructions is that when an alarm is detected, the controller will publish the event to FactoryTalk View Alarms and Events servers that propagate alarms to FactoryTalk View SE clients (HMIs) that subscribe to receive notifications.

The Digital Alarm (ALMD) instruction detects digital alarms based on Boolean (true/false) conditions. The instruction is an output instruction that obtains its alarm condition from the rung condition. Some of the features of the ALMD instructions include alarm acknowledge, latch, minimum duration timer, alarm counter, time stamp, severity, etc.. Figure 23-13 Rung 1 shows and example of the ALMD instruction.

The Analog Alarm (ALMA) instruction detects analog alarms based on the level or rate of change of an analog value. The instruction is an output instruction and is typically placed on an unconditional rung. Some of the features of the ALMA instructions include alarm acknowledge, high and high/high limits, low and low/low limits, dead band, rate-of-change, minimum duration timer, alarm counter, time stamp, severity, etc.. Figure 23-13 Rung 2 shows an example of the ALMA instruction.

Because of the many features of both of these instructions it is not practical to include a complete description here. Please refer to the manufacturer’s instruction manual for a complete description.
Chapter Summary

Many PLC applications require some form of alarming when abnormal conditions occur. When adequate alarming is provided, machine or process down-time can be minimized which in turn increases productivity. When programming alarm logic, the programmer should take into consideration how alarms will be captured, annunciated, acknowledged, and reset. In small PLC program applications the alarm logic can be programmed in conjunction with the logic that controls the machine or process. On the other hand, if the PLC program is large with many alarms it may be easier to create dedicated alarm logic to process and update all alarms at one time which will save time, increase PLC scan time, and decrease memory usage. It is not uncommon to find PLC programs that have been created with dedicated program files for just handling alarms.

Most HMI systems today have extensive alarming and event capturing features that can reduce or eliminate the need for PLC programmed alarm logic. These HMI systems can be programmed to monitor the PLC tag and data files that contain the machine or process information such as pressure, level, running, fault, etc.
**Review Questions**

1. What pair of PLC instructions can be used to capture and retain alarms?
2. How does the logical XOR instruction work?
3. Write the alarm logic that will alarm if a process temperature should rise above 250° F or fall below 100° F.
4. In Figure 23-12 what would you change to increase the number of alarms to 128?
5. In the PLC logic below, when will the flow alarm be active?