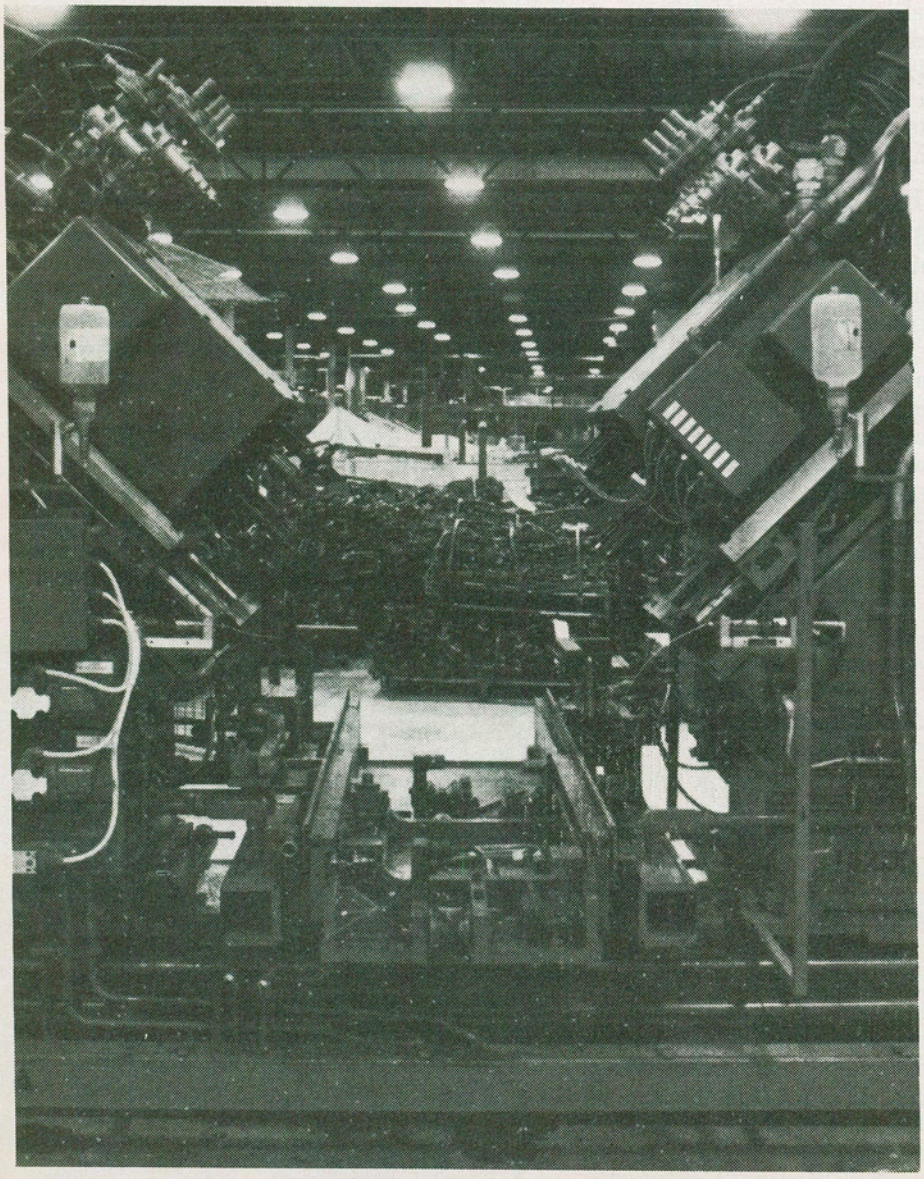
Safety First For Industrial PC Use

It pays to play it safe.

DR. H. VIRANI



Industrial machines of this type can function safely and efficiently through programmable controllers.

Programmable controllers (PC) are so easy to program and use that safety aspects, which would never be overlooked in a hardware control system, are sometimes never addressed. This article outlines some methods of increasing the safety of programmable controllers, controlled machines, programming techniques, physical installation practices, software diagnostics, maintenance considerations, and human factors that can help reduce the possibility of errors.

Economics of Safety

There are many reasons for being aware of safety in PC based control systems - but people have to be on the top of the list!

Preventing even one injury, or worse yet, a death, should satisfy even the most hard hearted that the small extra investment is justified. Also, the cost of damaged or spoiled products, due to a machine error far outweighs the cost of thinking ahead and preventing at least the obvious errors. Lastly, preventing damage to the machine being controlled by stopping dangerous motions before they cause problems will provide more up-time and a lower cost per unit for manufacture. Thus, better safety not only prevents injuries to people, it helps the entire system operate more efficiently.

The cause of unsafe control systems and machines are as varied as the machines themselves. But one of the major contributors is the ease with which a modern PC can be programmed. When PC manufacturers make it easy for designers to enter logic into the machine, they also make it easy to enter mistakes. It doesn't take much thought and effort to get the system up and running, and this simplicity tends to make the designer overlook things he would normally consider.

The ease of programming brings up another problem - that of unauthorized personnel making changes to programs. Many maintenance personnel and operators are allowed access to a PC; while they possess excellent job skills in their field, many lack the knowledge to program a PC safely. Even persons who do know how to program a PC may not know the procedure followed in your plant, and may make program changes that function properly but lack safety considerations. In some plants, Prom or Eprom memory for program storage might be the only protection against unauthorized tampering. Consider such a feature when deciding which PC to use for a particular application.

Dealing With Outputs

The primary techniques for increasing safety lie in the coding of the program itself. One of the most abused is the latched output i.e. an output that stays on once it is been activated. Latches are great for storing error codes, but problems arise when these latched coils are used to control "real-world" outputs. This is because conditions which make a particular output necessary during a machine cycle can change while the power is off. The latched output, however, will still be on regardless of the input conditions, possibly causing a serious or dangerous machine motion to occur immediately upon power-up. All outputs that cause any machine movements should be programmed with non-retentive outputs. If it is necessary to hold them up, use a hold circuit and seal in contacts just as you would with relays.

Most PCs provide I/O cards with isolated inputs and outputs. If you are controlling motor starters in a remote motor control centre, use isolated 120 volts outputs. You can never tell what phase is going to be used at the remote area, or where the power is going to come from. It pays to play it safe. Also, if the remote motor control centres have ground fault interrupt (GFI) protection (and they should), consider monitoring the state of the ground fault interrupt device in the main control program. When interfacing your system to a remote machine, use isolated I/O to prevent ground loops, common mode voltages, shocks, and other electrical problems. Use reed-relay contact outputs and isolated inputs, and make all interface wiring yellow or some other distinctive color for easy identification.

To decide how to specify limit switch wiring, remember that they are not all necessarily normally open (NO) devices. Use NO or NC (normally closed) switches for the greatest safety. For example, if the limit switches are on a hydraulic cylinder, decide which switch covers the most unsafe position, and make that switch normally closed. The most common failures are either a broken wire or an unmade contact, so if the wire or switch fails, the PC will sense an unsafe condition. If you wish to have all your inputs consistent within the program, simply invert the sense before using them. In a PC with an Exclusive OR data function, a complete word of such inputs can be inverted by Exclusive ORing with all ones (Fig.1).

Fig.1

```
Input data word: 11001100 11100111
XOR with all 1s: 11111111 11111111
                 --------------------
                 00110011 00011000
```

If this is not available, energize an internal output with the "off" limit switch input, and then use that output as the limit switch in the program. In very sensitive areas, where safety is of extreme importance and both positions of the cylinder are potentially unsafe, wire both the NO and the NC contracts to two separate inputs.

When using these inputs, insure that both are in their correct state before allowing the machine to proceed. Examine all the outputs carefully with regard to the PCs "last state" function; that is, what does the PCs I/O rack do to the output signals if fault occurs? Turn them off? Maintain the last output state? In the case of a CPU fault or power surge, not all outputs are safe when turned off. A pump maintaining hydraulic pressure, for example might be safer left on rather than turned off. If the PC doesn't maintain the last state, consider the use of "Rack Fault" switches. Here, a separate smaller rack is used for outputs that need to stay in their last state during a fault. In any case, don't just blindly assign everything to shut off after a failure.

Don't Trust the Hardware

When controlling anything that can do harm to the product, to the machine, or to a person, don't control that motion with a timer. Mechanical parts can slow down or speed up, but timers don't re- adjust to keep up. Also don't assume that a motion occurred because you sent an output to cause it. Use a limit switch input to verify that the movement did in fact happen.

If photocells or proximity sensors are used, make them failsafe, just like the limit switches. Photo electric sensors should be designed into the machine so that the dangerous condition breaks the beam. If the bulb burns out, the unsafe condition will be indicated. When you are unable to arrange this, write the program to check the photo cell when it is illuminated, at least once every cycle. In very sensitive situations, use a current sensor in the lamp or LED circuit to monitor the light. Proximity sensors, likewise, should provide an off signal when the unsafe condition is present. These, too can be checked during another part of the cycle to verify proper operation.

If a conveyor belt or pump must be on for safe operation, use some sort of feedback to make sure. This can be as simple as an auxiliary contact on the motor starter. Belt motion can be sensed by a tachometer. A running motor can be sensed by current sensors in the lamp or led circuit to monitor the light.

Installation Aspects

When grounding control elements, read and follow the manufacturer's recommendations. Make sure all parts of the PC, the power supplies and the cabinets are grounded with heavy gauge copper wire. Use the necessary power conditioning on all ac lines and install transient protectors across motors, solenoids and other inductive components that are switched by "hard" contacts. Noise on the ac lines can cause spikes on the power supply, and spikes on the power lines can cause intermittent errors in the logic. And logic errors can cause the machine to destroy parts, waste products, and even kill or maim personnel.

Don't let the "magic" of the PC make you forget that some things must be hard wired. For example, all emergency off actuators and circuits need to hard wired. Make sure that when power is cycled off and on the control relay doesn't pick up by itself. Wire your circuit, so that a manual reset is required before power returns to the machine.

Diagnostics for Safety

Most PCs use an internal "watchdog timer" for monitoring internal functions, and they belong in your program also. Overall cycle time of the machine can be monitored with a timer preset for about 120% of the normal time. Reset the timer once each cycle, and if the time is ever out, it will indicate something is wrong (hopefully before an accident occurs). Many machine motions can be monitored with one timer set up as a master clock. As each motion finishes, compare the time with a stored valve, and set off an alarm if the time exceeds the norm. A 10% to 50% leeway should be allowed, depending on the safety impact of each motion to prevent nuisance alarms.

Data comparison is very powerful diagnostic tool. For example, inputs can be examined during a portion of the machine cycle when conditions are known and static. This method lets many sensors malfunctions be detected before they cause trouble. Discrete I/O points can be checked one at a time, or whole input

Safety First For Industrial PC Use

words can be checked. For this to be really efficient, inputs can be grouped on the I/O modules during the design stages.

Maintenance Considerations

Many PCs allow "forcing" inputs and outputs (that is, an input or an output can be turned on or off manually, regardless of process conditions). Forcing I/O is often done during maintenance or troubleshooting to simulate inputs or to bypass certain portions of a program. Usually, forcing is allowed only in one section of the I/O rack at a time.

Using Extreme Caution With Such Features

Forcing I/O in one section can cause outputs previously forced off in another section to come on at once, with disastrous results. Again, by anticipating for safety, you can group all related I/O in one section to allow future forcing without causing unexpected results. In a conveyor system, all stops could be wired to the same input group; this allows forcing the I/O to manually run a load through the system during troubleshooting. Other problems occur when you alter a program while the machine is running.

Even changing the preset of a timer at the wrong time can cause unexpected actions to occur. On-line programming is especially dangerous. Use the CPU key lock provided by the manufacturer or install your own to prevent indiscriminate on-line program changes by maintenance personnel. PROM program storage will be the best answer to the problem.

Keep the maintenance function in mind during all your design plan. Give technicians overrides for watchdog timers and safety circuits where necessary - if you don't, those safety circuits may mysteriously disappear from the program. Maintenance overrides, while necessary, can be misused.

Consider Operator Needs

Estimates are that 40% to 50% of the reliability of a system is attributable to its operators. Thus, it pays to consider them during the design. Get input from the people who will operate the machine on matters such as placement of alarms and indicators. The operator must be able to understand the controls and react to alarms in a manner and time frame sufficient to prevent accidents. If operators help design the console, they will be able to run it better.

Remain consistent with terms and abbreviations throughout the machine and within the organization. Colors should have the same meaning everywhere and emergency conditions should always be shown in the same manner. Emergency stops must either always turn on, or always turn off the stop lights - not some on and others off. Alarms should indicate severity by placement, color, and even sound, if applicable.

Minimize False Or Meaningless Alarms

Lights that are almost always on are often ignored in an emergency. Provide a means to override any noncritical alarms during maintenance activities to give more importance to serious alarms. In any case, don't allow any inactive alarm indicators to remain on. For maximum effectiveness, every alarm the operator sees should be real. Set the trip levels at reasonable levels to minimize unnecessary activation. If necessary, consider providing two alarms levels - yellow and red, for example. Requiring a reset button to be pushed twice in the event of a serious fault can prevent accidental machine restarts that could have disastrous results.

If your console has some type of machine parameter display, use a logical arrangement so that relationship to the machine is obvious. Where many conditions must be displayed at the same time, use methods to assist the operator in understanding them. Wherever possible, let the PC, not the operator, verify that necessary actions have taken place. For instance, if the operator requests a pump to start, program the PC to check after a reasonable delay to verify the starting and alert the operator only in case of a malfunction. If you can reduce the operator workload by extending this concept to all important movements, you can allow him or to go about their job more efficiently and safely.

Safety After Start Up

Most systems using programmable controllers have a surplus of memory available after the machine control program has been written and debugged. Many times extra memory is included just because " we might need it". Use some of this memory to provide diagnostic and safety features to enhance the rest of your safety-oriented design.

Guard against adding so much protective gadgetry that the original function of the machine is encumbered. Safety functions must complement, not get in the way of the real purpose of the machine. Those that do get in the way will eventually be removed, leaving a machine with no protection for itself, or the people and products around it. ■