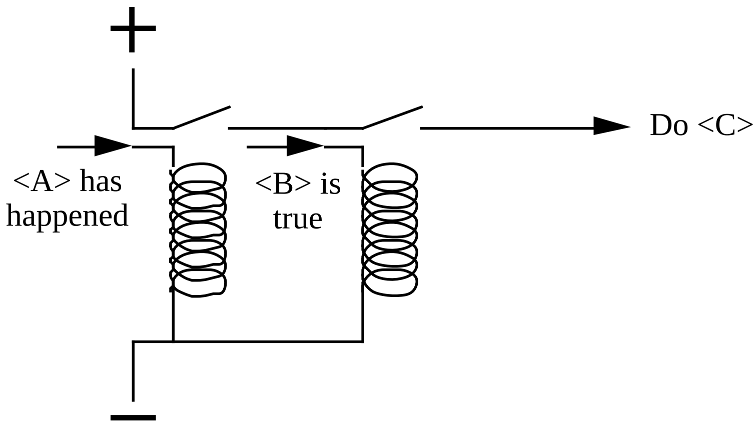
Computer Science 773

Robotics and Real-time Control

# PROGRAMMABLE LOGIC CONTROLLERS

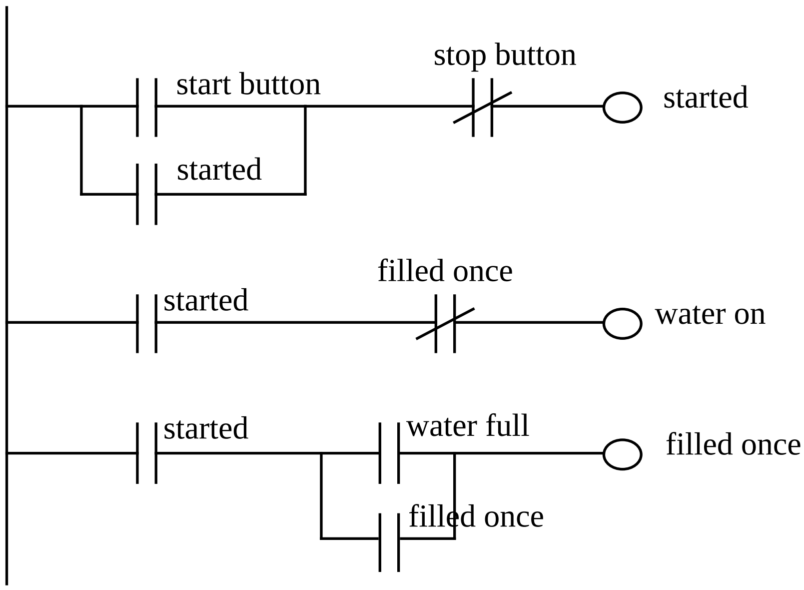
A very large number of industrial processes are controlled by *sequencing.* They are composed of a number of operations, and the main control activity is to ensure that these operations follow one another in a well defined sequence. The typical control logic is "if <A> has happened and <B> is true then do <C>", which is all very Boolean and binary.

In earlier times, this sort of control was dealt with by building circuits of relays. Here's a very simple circuit which implements the instruction above :




Beginning with two positive signals representing "<A> has happened" and "<B> is true", both relays are energised, both switches close, and the result is a positive signal representing "do <C>". Notice that the result is in just the same form as the original inputs, so it can be used for further computations if you so wish.

It all works, and it's rather reliable. People built very big control systems using this sort of relay logic, and they worked well. This became sufficiently standardised to have its own programming language, called *relay ladder logic* ( or some variant thereof – you find slightly different names in different places ). It's a graphical parallel declarative language, and it started well before computers became common. Here's an example of a *ladder diagram* :



The logic is arranged in "rungs" of the "ladder", with each rung computing one value. The top rung sets a *started* signal whenever the *start button* is pressed, and holds it set until the *stop button* is pressed. If the system is *started* and hasn't been *filled once*, the water is turned on; when the *water full* detector is activated, the *filled once* signal is turned on ( thereby stopping the flow of water ) and held until the *stop button* is pressed. And so on.

Relay systems are reliable, but expensive, and bulky, and hard to rewire if you want to change the plant. Nowadays, these hard-wired systems have been replaced by programmable logic controllers ( PLCs ) which are built on microcomputers, and simulate the same functions with software. Ladder diagrams are still in use, though other notations which usually look more like programming languages are becoming more common.

A PLC is usually a microprocessor running a simulator of the sort of relay network described above. It might also run a user interface programme with which one can enter new networks using ladder diagrams or otherwise, and monitoring and testing software for programme development and fault diagnoisis. In some systems the user interface and testing software run on separate machines, as the PLCs themselves usually run unsupervised with no intervention for long periods.

It will also be equipped with a range of interfacing devices, usually connected to the microprocessor by a common bus. These typically come with the PLC package, and you orde those you require for your own installation.

PLCs are only now, and rather slowly, being superseded by other ways of achieving much the same ends :

http://www.cutlerhammer.com/op_a_sys/so_ne_so.html

NetSolver

NetSolver is a complete software package that enables an industrial personal computer (iPC) to function as a full feature PLC, operator interface panel, and PLC programming terminal.

NetSolver allows the user to:

* develop and run a control system with graphical flowcharts
* control I/O directly from the iPC (eliminating the necessity of a PLC)
* develop and run an operator interface
* share data with other Windows-based programs through Dynamic Data Exchange (DDE)

Features and benefits of NetSolver:

* deterministic control operating system
* windows user interface
* optional integrated MMI
* modular, easy to use editors

# APPLICATION NOTE

---

## COMPRESSOR CONTROL

1. REQUIREMENT

It is required to load and unload Howden compressors in response to changes in suction pressure. The load and unload signals are pulsed outputs, and have the following typical mark/space ratios :-

LOAD : 1 second ON - 9 seconds OFF

UNLOAD : 2 seconds ON - 8 seconds OFF

The suction pressure is available as a conventional transducer output, as a 4-20mA, 0-5 volt, etc. range. The following are typically required responses, given in terms of transducer voltage outputs :-

| SUCTION PRESSURE VOLTAGE | OUTPUT |
|---|---|
| 0.2 to 2.4 volts | LOAD |
| 2.41 to 2.59 | DEAD BAND ( NEITHER LOAD NOR UNLOAD ) |
| 2.6 volts + | UNLOAD |

2. IMPLEMENTATION

The attached ladder diagram shows one method of implementing this requirement, where :-

| | | |
|---|---|---|
| OUTPUT 14 | = | LOAD |
| OUTPUT 15 | = | UNLOAD |
| INPUT 31 | = | SUCTION PRESSURE VOLTAGE |

Line 1 shows the suction pressure voltage being taken in and stored at location 400. If present, value is 28 "units" ( 2.8 volts ).

Line 2 shows the voltage being compared against the low end of the dead band : if higher than 2.4 volts, operate relay 200.

Line 3 shows the voltage being compared against the high limit of the dead band : if higher than 2.6 volts, operate relay 201.

Lines 4, 5, 6
show the required combinations being derived :-

Relay 202 energised = Load required;
Relay 204 energised = Unload required.

Line 6 shows a self-resetting timer ( 10 secs ) that pulses at 1 second intervals.

Line 7 shows a technique for storing the current count ( 4 ) of the timer in location 417.

Lines 8 & 9
cause the Load and Unload pulsing to be the correct mark/space ratio by comparing the timer count with 9 ( for Load ) or 8 ( for Unload ). Relay 202 enables the Load ( 14 ) pulsing output while relay 204 enables the Unload ( 15 ) pulsing output.

APPLICATION NOTE

---

COMPRESSOR CONTROL

```
INDUSTRIAL SOLID STATE CONTROLS                                      MODEL 395 L0/DER/MONITOR

                                   LADDER LISTING

LINE
NUMBER

   0  ! BEGIN OF MEM
      !
   1  !   31                                                                          400
      !--[D]--+---------------------------------------------------------------------(S)--
      !   12                                                                           12
      !
   2  !  400                                                                          200
      !--[B]--+--]G[--+--[K]--+-----------------------------------------------------( )--
      !   12              24
      !
   3  !  400                                                                          201
      !--[B]--+--]G[--+--[K]--+-----------------------------------------------------( )--
      !   12              26
      !
   4  !  200     201                                                                  202
      !--]/[--+--]/[--+-------------------------------------------------------------( )--
      !
   5  !  200     201                                                                  204
      !--] [--+--] [--+-------------------------------------------------------------( )--
      !
   6  !               +-----+                                                         206
      !---------------! TRO !-+-----------------------------------------------------(S)--
      !               !     !
      !  206          !     !
      !--]/[--+-------! RST ! PRS    10
      !               !     ! ACC     4
      !               !     !
      !               !     !
      !               +-----+
      !
   7  !                                                                               417
      !-----------------------------------------------------------------------------(S)--
      !                                                                                 4
      !
   8  !  417                     202                                                   14
      !--[B]--+--]G[--+--[K]--+--] [--+---------------------------------------------( )--
      !    4               9
      !
   9  !  417                     204                                                   15
      !--[B]--+--]G[--+--[K]--+--] [--+---------------------------------------------( )--
      !    4               8
      !
```



Alan Creak,
March, 1998.