Control of a Large Process Using a Small PLC

RAMADAN A. FAN
Systems Engineer
Saudi Aramco
Dhahran, Saudi Arabia

L. CHEDED
Assistant Professor
Systems Engineering Department, KFUPM
Dhahran, Saudi Arabia

ABSTRACT

The objective of this paper is to demonstrate how a programmable logic controller (PLC) with a limited input/output capability, such as the 12-input/8-output Omron PLC, coupled with a low-complexity interface circuit can be used to control a large process. The test bed for this process is a traffic light system whose number of outputs (26) far exceeds the 8 outputs available from the PLC. The key and novel idea behind this successful PLC-Interface combination resides in the clever coding of the various states that the process under control has to go through. An illustrative low-cost demonstration system was recently successfully tested at the Systems Engineering Department. Finally, the coding technique used here can also be extended to other similar large processes.

1. Introduction

Programmable logic controllers (PLCs) are becoming more and more popular in industry due to their flexibility, reliability, and cost-effectiveness. Their architecture is similar to that of a conventional digital computer, consisting of a central processing unit, a memory unit, and a number of input/output (I/O) terminals. The strength of a PLC, however, lies in its optimized manipulation of one-bit logic operations. The program is scanned continuously to see if certain conditions on the inputs are met so that appropriate actions on the outputs are taken.

One of the major concerns when selecting a PLC for an application, besides speed, is the number of input/output (I/O) connections it can provide. PLCs come in different sizes and with varying capabilities, ranging from micro PLCs that have up to 32 I/O connections and 2K words of memory, to large ones with an expandable I/O capability of up to 8192 I/O connections and 4M words memory [3].

The choice of a PLC appropriate for a particular control task is usually determined by analyzing the process to be controlled, and finding out the number of I/O connections required for the interface. As expected, the required number of I/O connections will increase with the complexity of the control task at hand, and a bigger PLC will have to be used, hence raising the cost of the overall system.

A question that naturally arises at this stage is the following: "Can a PLC with a modest number of I/O terminals be used to control a relatively sophisticated process?" An affirmative answer to such question would certainly be of great educational and practical benefit. The feasibility of this idea was tested in a senior-project course in the Systems Engineering Department at KFUPM, where the process chosen was a traffic light system providing both dumb and intelligent control. This paper presents both the hardware and software aspects of the successful design and operation of this PLC-based controller.

1.1 PLC-Process Interface

Usually, the interface between a PLC and a process begins with an analysis of the number of I/O connections required. This will determine the size, type and cost of the controller. But what if we were restricted to a PLC that has only a limited number of I/O connections, can we still use it to accomplish the desired control task? One solution to this problem, as presented in this paper, would be the introduction of an intermediate stage which will interface the PLC to the complex process.

We propose here an encoding/decoding approach that would expand the fan-in and fan-out capabilities of the small PLC used. The argument behind this is that, although PLCs were originally developed to supplant hard-wired controllers, a hybrid solution involving a PLC and some hard-wired interface may entail an increase in complexity but would, in return, offer a cost saving.

Considering the time frame of our project, only one part of the problem was investigated, i.e. extending the fan-out capability of the PLC used. A digital decoding circuit was designed to interface the PLC outputs to a traffic light system. We assume that the same approach can be used to extend the fan-in capability of the PLC, leading to the design of an encoding interface circuit between the traffic lights and the PLC inputs.

1.2 Design of the Experiment

Figure-1 shows the four-way isolated intersection used in this project. The major road has separate signals for "straight-forward" and "left-turning" traffic, whereas the minor road has a common signal for both. The push buttons represent pedestrian and vehicle sensors. The result is a total of 26 lights and 10 push buttons. The PLC used is the Omron Sysmac C20K, with 12 inputs and 8 outputs [4]. Although the inputs are sufficient for direct connection with the push buttons, the outputs are not enough for the traffic lights. Therefore, we decided to consider the 8 PLC outputs as an 8-bit code string, that will be decoded by a digital decoder into the required number of outputs.

But the question is how to encode the various traffic light states. A literature survey on traffic control schemes [5, 6] showed that there are eight basic states which the controller has to switch between. Figure-2 shows these states arbitrarily numbered from 1 to 8. Based on the fact that there are extra intermediate states, such as the green/yellow and the all-red states, between the above-mentioned eight states, it was decided to use some additional "control signals" for a proper sequencing of the traffic lights.

The project was designed to carry out the control in three different modes: dumb control, intelligent control, and flashing mode. These three modes are programmed in the PLC and their selection is made from a PC which acts as a master controller. The complete setup is shown in Figure-3, where the feedback loop connects the ten push-buttons of the traffic light circuit back to the PLC input section. The remaining two PLC inputs are used by the PC for mode selection according to the following codes:

0 0 *Flashing:* Used for emergency repairs at the intersection or for signaling PC breakdowns.

0 1 *Dumb Control:* Control is done by outputting pretimed sequences.

1 0 *Intelligent Control:* Control senses requests from pedestrians and vehicles which will interrupt the normal sequence.

1 1 *All-Red:* Used as an intermediate state for safe transitions between consecutive modes.

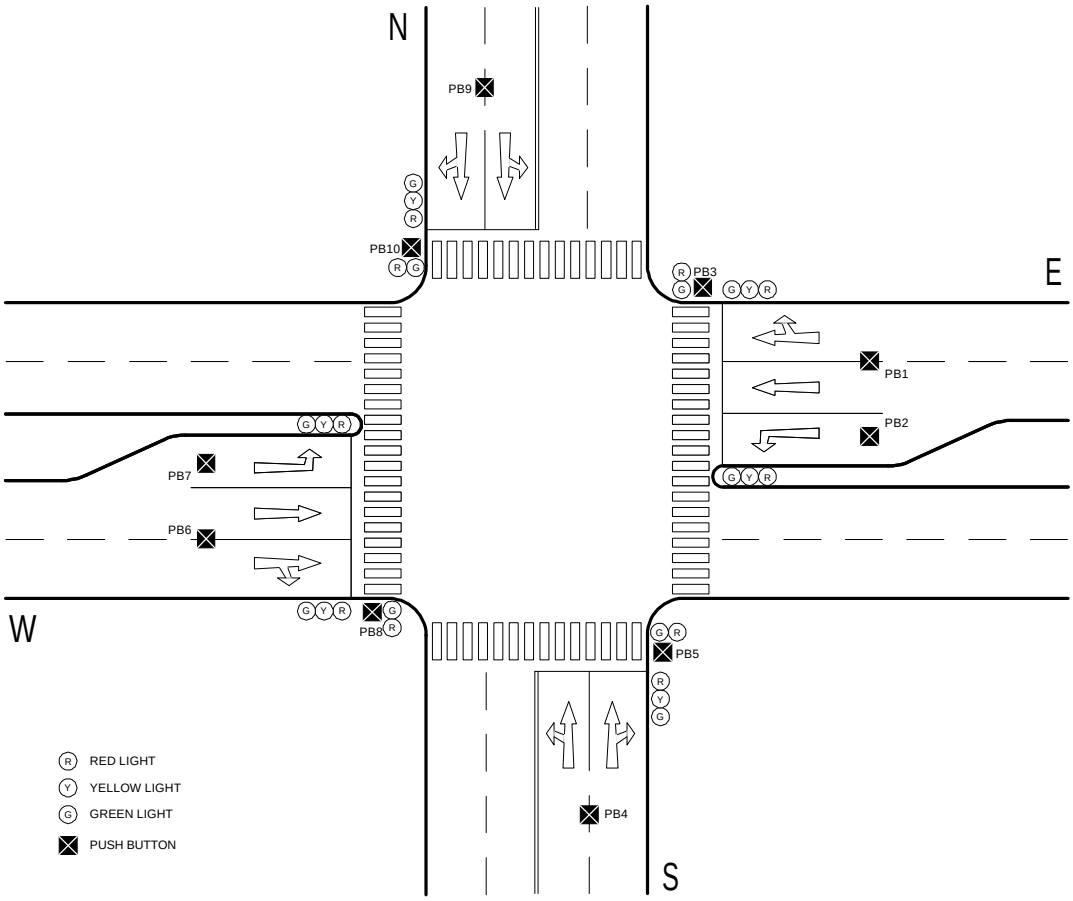


Figure-1 The four-way intersection between a major road and a minor road.

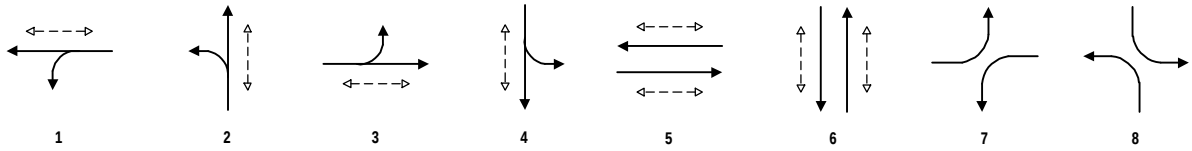


Figure-2 The eight basic states for traffic light control (arbitrarily numbered).

2. Hardware Design

2.1 PLC Output Interface

The eight PLC outputs, numbered here by their addresses (100-107), are assigned as follows:

Address	Label	Meaning
100	VG	Vehicle Green Light
101	VY	Vehicle Yellow Light
102	PG	Pedestrian Green Light
103	E	Emergency Signal
104	A	3-bit State Code
105	B	
106	C	
107	ENB	Buffer Enable Signal

The first four outputs are used as control signals to differentiate between the intermediate states. The red lights are not included since they are always the complement of their corresponding green lights. The next three outputs (A, B, C) are used for addressing the eight states, and the last output (ENB) is used for buffering purposes. Each state consists of three consecutive sub-states: all-red, green, and green/yellow. The flashing mode has two sub-states: all-red (for 5 seconds) then flashing. Whenever one of these states needs to be activated, the PLC program will output the required sequence of bits.

Figure-4 shows the design of the decoding circuit used. First, an octal buffer is used to latch and synchronize the eight PLC outputs. Then, a 3-to-8 decoder is used to decode the 3-bit state code into one of the eight basic states. The combinational logic circuit will then turn on only those lights corresponding to the active state. Finally, the timer is used to provide the decoding circuit with its flashing capability.

Several assumptions were made in order to simplify the design. We assumed that there are no separate right turn signals in either road. Also, the minor road has no separate left turn signals. In effect, states 6 and 8 (of Figure-2) were not used because they require that the minor road have separate left turn signals.

One limitation of our decoding circuit is that some transitions are not possible, for example, between states 1 and 5. During this transition, the straightforward signal should stay green while the left signal changes from green to green/yellow to red. The way the circuit is built requires that *all* green lights go through the intermediate sub-states before entering the next state. The solution is to either add these green/yellow sub-states to the eight basic states and use a larger decoder, or simply avoid transitions such as 1-5, 3-5, 1-7 and 3-7 in the control. The latter option was chosen to keep the overall design simple as it was meant for illustration purposes.

2.2 PLC Input Interface

The PLC inputs are connected as follows:

Address	Label	Meaning
000	S0	Mode Selection Bit
001	S1	Mode Selection Bit
002	PB1	East-Straight Vehicle Push-Button
003	PB2	East-Left Vehicle Push-Button
004	PB3	East Pedestrian Push-Button
005	PB4	South Vehicle Push-Button
006	PB5	South Pedestrian Push-Button
007	PB6	West-Straight Vehicle Push-Button
008	PB7	West-Left Vehicle Push-Button
009	PB8	West Pedestrian Push-Button
010	PB9	North Vehicle Push-Button
011	PB10	North Pedestrian Push-Button

The twelve PLC inputs were sufficient for our application. Recall that two of these were used for mode selection, and the remaining ten for push-button connections. But had their been more push-buttons, a hardware solution could be found by analyzing the requests and seeing which ones could be ORed together, or multiplexed by hardware and then demultiplexed by software, which is the reverse of what we have done in the output decoding circuit. The extension of the PLC fan-in capability was not included in this project.

2.3 PC-PLC Interface

The role of the PC in this project is limited only to selecting between the three different control modes. It has no influence whatsoever on the traffic light control itself. In fact, Figure-3 shows that the PC is indeed outside the control loop. The PC merely simulates a remote master controller which uses the PLC as a slave.

3. Software Design

3.1 Control Logic

The coding of the control logic is accomplished on two different platforms. The first is the PC which uses a BASIC program to provide the user interface and mode selection capability. The other is the PLC which uses ladder logic to activate the desired control mode. The communication between the two programs is done through the S0 and S1 signals.

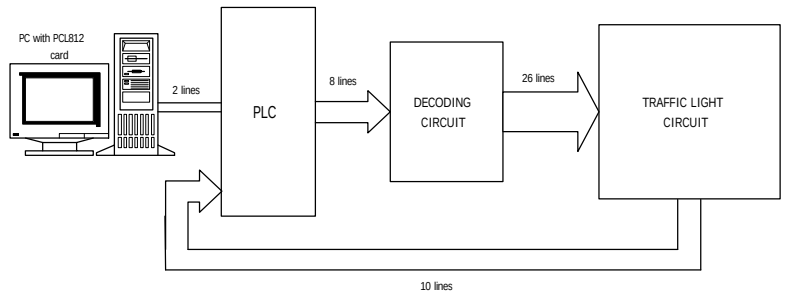


Figure-3 Block diagram of the traffic light control system used in the project.

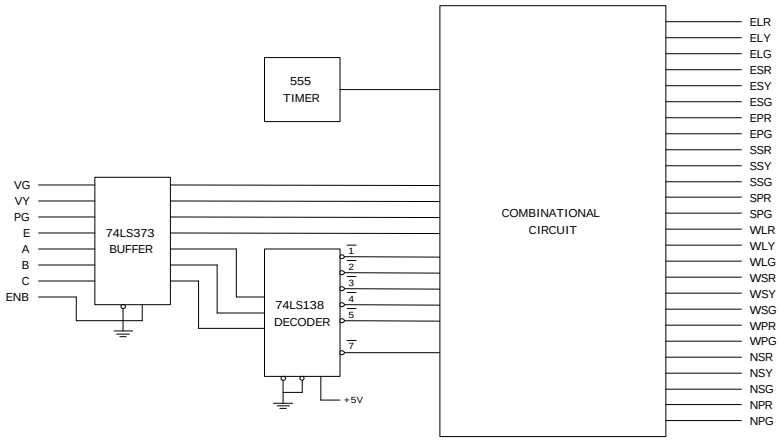
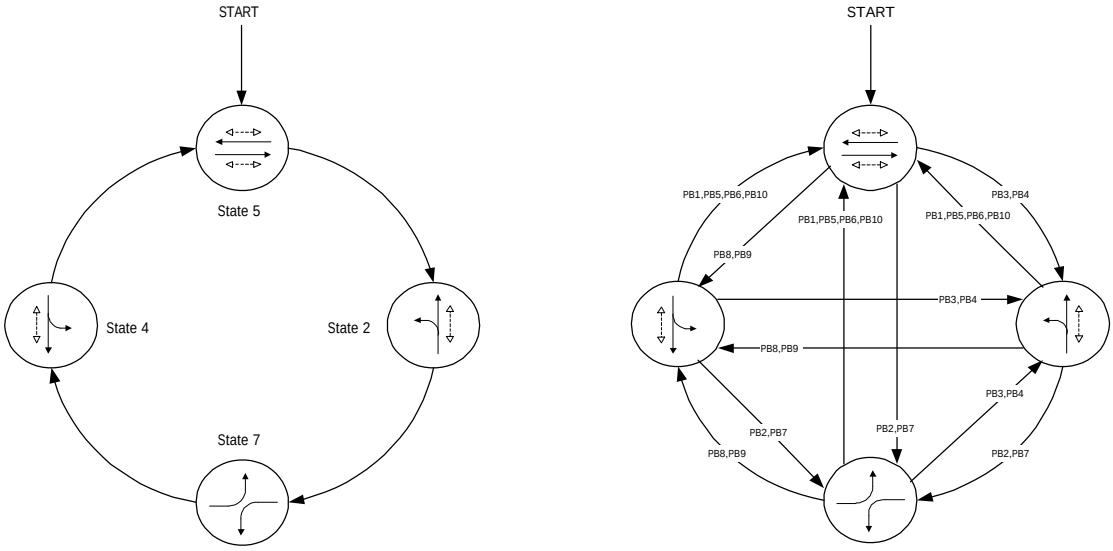


Figure-4 PLC output interface using a decoding circuit.



(a) **(b)**

Figure-5 State diagram for (a) dumb control (b) intelligent control.

Figure-5 (a) shows the state diagram for the dumb control mode. The system has to go through a sequence of four consecutive states: 5-2-7-4, Each consisting of three sub-states: all-red, green, and green/yellow. The outputs that should be generated and their timing sequence are shown in Table-1. Notice that we gave state 5 the longest total duration since it corresponds to the major road.

Table-1 also shows the timers that are associated with each output in the PLC program. Whenever an output needs to be turned on for the corresponding duration, a combination of internal timers and latches are used. For example, the VG output is to be kept high in state 5 for the duration of 20 secs. (= 15 + 5) and in states 2, 7 and 4 for an equal duration of 15 secs. (= 10 + 5). Hence for the VG output alone, four timers are used (TIM0, TIM1, TIM2 and TIM3), each of which is kept on for the prescribed duration by its own latch.

Figure-5 (b) shows the state diagram for intelligent control. In the absence of any interrupts from the push buttons, the intelligent control mode is the same as the dumb control mode. This is indicated by the circular transition lines, which are similar to those of the dumb control. However, if an interrupt occurs, the control will activate the corresponding state. This is indicated on the state diagram by the added transitions in the middle. The timing of each state remains the same, but the sequence of states may be different.

A situation may arise when several interrupts occur simultaneously. The ideal solution is to service these interrupts based on a first-come-first-served basis, or based on the relative importance of these requests. In this project, however, there are no priorities defined. The program simply activates whichever state it scans first in the ladder diagram.

If the flashing mode is selected, then the system will first stay in the all-red state for 5 seconds (for a safe transition from a currently active state), then starts to flash. If the master controller (PC) suddenly fails, the PLC will automatically turn the flashing signal on because the S0 and S1 inputs will be low. The required output sequence is shown below in Table-2.

3.2 Ladder Implementation

The implementation of the control logic described above is accomplished using ladder logic. The first part of the ladder sets the control in the desired control mode. This is done by scanning the PLC inputs 000 and 001. If the dumb or intelligent modes are chosen, the control will start from state 5 as indicated on the state diagrams of Figure-5. The program then continues to scan the PLC inputs during the activated mode in order to detect any change in the desired mode, or in the case of intelligent control, detect requests from push buttons. Based on these inputs, a certain state can be entered and the corresponding outputs can be activated.

For example, in order to enter state 5, the program has to sense either a dumb mode start up, an intelligent mode start up, a push button depression, or simply the end of the previous state (state 4) if it is in normal operation. The first output to be activated is “C” as indicated in Table-1, initially for a duration of 5 seconds (TIM18) then for 20 seconds (TIM19). The reason for using two timers here is that the end of TIM18 is used to send a signal to two other timers, namely TIM8 (for activating “PG”) and TIM0 (for activating “VG”). Similarly, the end of TIM8 sends a signal to TIM4 for activating “VY”.

State	C	B	A	E	PG	VY	VG	Duration (sec.)
5	1 TIM 18	0	0	0	0	0	0	5
	1 TIM 19	0	0	0	1 TIM 8	0	1 TIM 0	15
	1	0	0	0	0	1 TIM 4	1	5
2	0	0	1 TIM 12	0	0	0	0	5
	0	0	1 TIM 13	0	1 TIM 9	0	1 TIM 1	10
	0	0	1	0	0	1 TIM 5	1	5
7	1 TIM 20	1 TIM 16	0	0	0	0	0	5
	1 TIM 21	1	0	0	1 TIM 10	0	1 TIM 2	10
	1	1	0	0	0	1 TIM 6	1	5
4	0	1 TIM 17	1 TIM 14	0	0	0	0	5
	0	1	1 TIM 15	0	1 TIM 11	0	1 TIM 3	10
	0	1	1	0	0	1 TIM 7	1	5

Table-1 Timing sequence for the PLC outputs in the four states (used in the dumb and intelligent modes).

C	B	A	E	PG	VY	VG	Duration (sec.)
0	0	0	0	0	0	0	5
0	0	0	1	0	0	0	Indefinite

Table-2 Timing sequence for the PLC outputs in the flashing mode.

An additional timer, TIM23 (not shown on Table-1), is used for activating the output "ENB" for a very small duration (0.5 seconds) in order to buffer the PLC outputs to the decoding circuit whenever there is a change in the state.

The procedure for encoding state 5 is duplicated for the remaining possible states of the system, with the exception of the startup events for both the dumb and intelligent modes. The complete ladder diagram was documented as part of the project report.

4. Conclusions

In this paper, the control of a traffic light system requiring more outputs than can be provided for by the small PLC used, was illustrated. It was shown that the solution to this control problem hinged upon the key idea of using the limited PLC outputs for the sequencing of the various states of the process to be controlled rather than for the direct control of the process outputs themselves.

This entailed the use of a software encoding scheme, followed by a hardware decoding circuit of modest complexity. We believe that this encoding/ decoding approach offers a good compromise between complexity and low cost on the one hand and design flexibility on the other, and as such, should therefore be attractive in situations where the cost of a larger PLC is prohibitive.

Finally, not only can this approach be used to increase the PLC's fan-in capability as pointed earlier, but it is also general in nature and hence, can be applied to other large processes with suitable modifications in the interface circuit.

References

[1] John W. Webb, Programmable Logic Controllers: Principles and Applications. New York: Macmillan, 1988.

[2] Ryan G. Rosandich, "What to Know About PLC Ladder Diagram Programming" EC&M Jun. 1996: 20.

[3] Kelvin T. Erickson, "Programmable Logic Controllers" IEEE Potentials Feb./Mar. 1996:14.

[4] Omron User Manual and Installation Guide, 1989.

[5] Charles V. Zegeer and Sharon F. Zegeer, Pedestrians and Traffic-Control Measures. No.139, Washington, DC: National Research Council, Nov. 1988.

[6] James H. Kell and Iris J. Fullerton, Manual of Traffic Signal Design. New Jersey: Prentice-Hall, 1982.