

## LABORATORY SESSION 1

### BASICS OF PROGRAMMABLE CONTROLLERS

#### PART I



Courtesy of Allen-Bradley

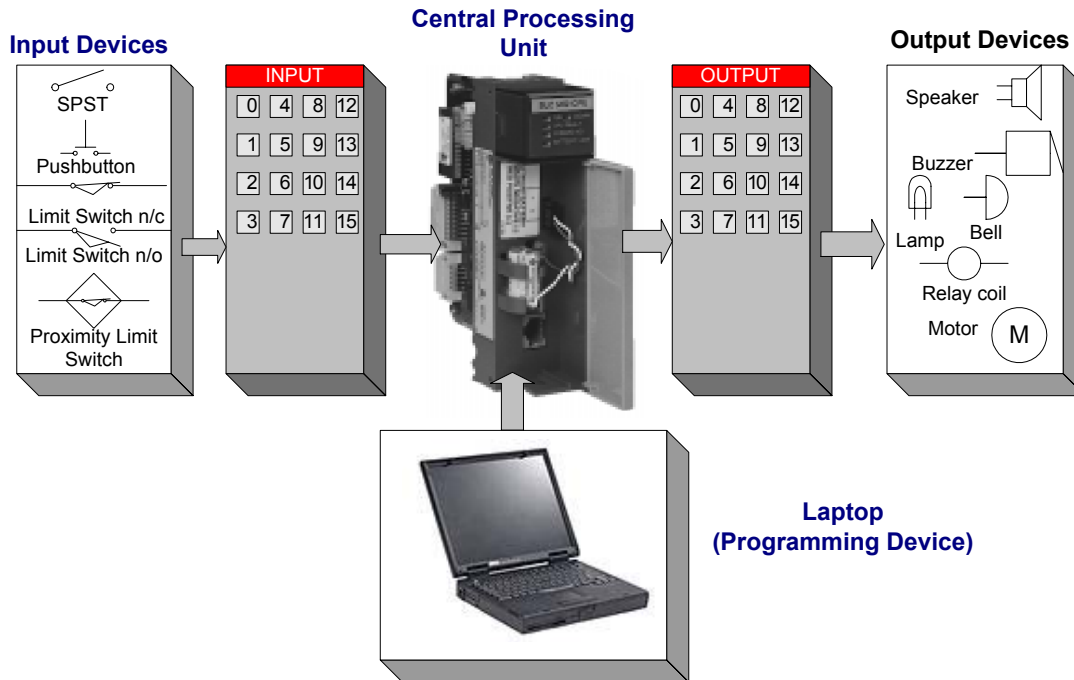
#### Purpose

1. To define what a programmable logic controller (PLC) is, and to identify the main parts of a PLC and describe their functions. To become familiar with the operation of the Allen-Bradley SLC-500 programmable controller.
2. To understand how the ladder diagram language is used to communicate information to the PLC.
3. To identify input and output devices including switches push-buttons, contactors and relays commonly used in PLCs.
4. To learn experimentally how a programmable controller is programmed. To describe the use of the RSLogix 500 software and to become familiar with some of its commands and capabilities for programming the Allen-Bradley SLC-500 Programmable Controller.
5. To program several simple ladder logic circuits, test them, download to SLC-500 and demonstrate their operation.

#### 1. Introduction

A programmable controller, also known as programmable logic controller (PLC), is a device designed to perform sequential logic control in which one event follows another in a prescribed way to complete a task. An industrial machine may contain pushbuttons, limit switches, timers, interlocks, etc. In the past, these devices had to be hard-wired to perform a specific function, and they could not easily be modified to perform a different task. With a programmable controller, however, all that is required is to change the controller's program and possibly some of the connections to the inputs and/or output. Today, programmable controllers are used for the control and operation of almost any machine, process, or production line. The continued advancement of PLC technology and the partnership of PLCs with computers and other digital devices are providing industry with sophisticated control systems and advanced decision making processes.

A programmable controller is composed of three components as illustrated in Figure 1.1. These three components are the input/output (I/O) interface system, the central processing unit (CPU), and a programming device.



**Figure 1.1 The Programmable Controller System.**

The input/output system forms the interface by which field devices are connected to the controller. The main purpose of the interface is to condition the various signals received from or sent to external field devices. Incoming signals from sensors such as pushbuttons, limit switches, analog sensors, motor starters, etc. are wired to terminals of the input interfaces. Devices that will be controlled, like motor starters, solenoid valves, pilot lights, and position valves, are connected to the terminals of the output interfaces.

The CPU section of a PLC has three components: The memory system, the processor, and the system power supply. The memory system stores the program usually in the form of a ladder diagram. The processor executes the control program stored in the memory system. The system power supply provides all the necessary voltages required for the proper operation of the various CPU sections.

The programming device allows the user to enter, change, or monitor a PLC program. The programming device may be a CRT terminal, a hand-held unit with a display or a personal computer. Four major languages can be used in programming the modern PLC. These languages may use Boolean algebra equations, mnemonic commands, logic diagrams and ladder diagrams. Some PLCs also permit BASIC language statements or allow the programmer to use a special high-level control language. Ladder diagram logic, which is the same as that for relay control circuits, is a popular choice for programming most PLCs.

During the operation, the CPU reads or accepts the input data or status of the field devices via the input interfaces, executes the control program stored in the memory system, and writes or updates the output devices via the output interfaces. This process of sequentially reading the

inputs, executing the program in memory, and updating the outputs is known as scanning. Allen-Bradley's programmable controllers are efficient and effective in providing industrial control. Allen-Bradley was one of the first companies to manufacture PLCs for enhancing plant productivity. Allen-Bradley's PLC family of programmable controllers, are among the most advanced and versatile PLCs worldwide. In the Industrial Control Lab we have SLC-5/02 in the benches and PLC-5 and SLC-500 on the wall.

### 1.1 Input-Output Devices

Both discrete and analog Input/Output devices are provided. Discrete input devices are essentially a switch that is either open or closed, signifying a 1 (ON) or 0 (OFF). The input panel on the bench shown in Figure 1.2; consists of the following devices:

- 8 Toggle switches (S0 - S7).
- 8 NO and NC pushbuttons (PB0 - PB7).
- 4 Thumbwheel switches (TW0 - TW3). Rotating switch to input numeric information into the controller.

Discrete input devices operate from a 120 V AC supplied from the back plane of the rack enclosure. (L1 - L2).

Discrete output devices provide connections between the programmable controller and output field devices.

The output panel on the bench shown in Figure 1.3; consists of the following devices:

- 8 Signal lights.
- 4 5-A relays with both NO and NC contacts (R0 - R3).
- 2 18-A, Contactors (C0 - C1).

The panel also contains analog input and output devices, which provide control capability for field devices, which respond to continuous voltage or current level.

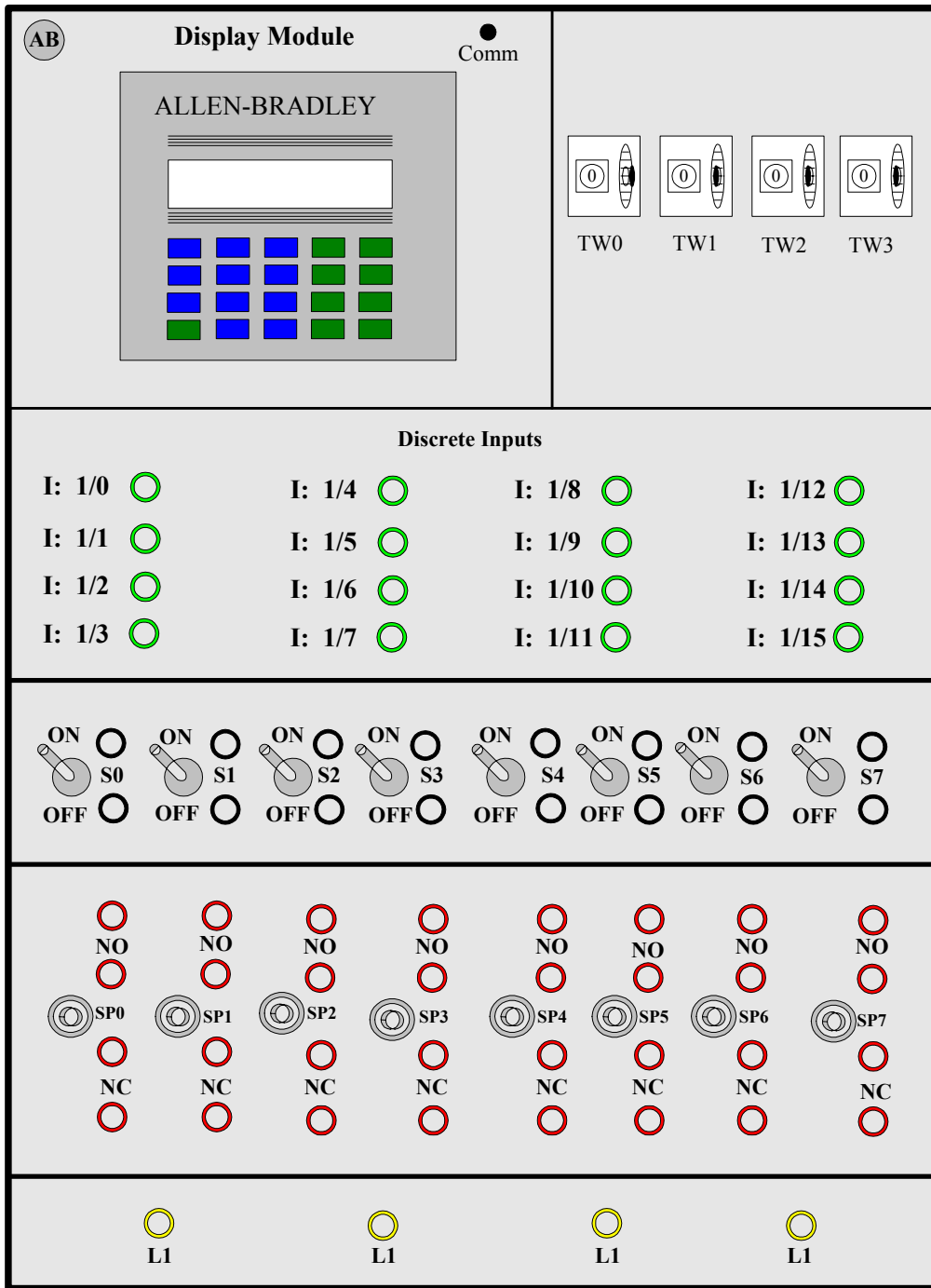


Figure 1.2 Bench Panel Input Devices

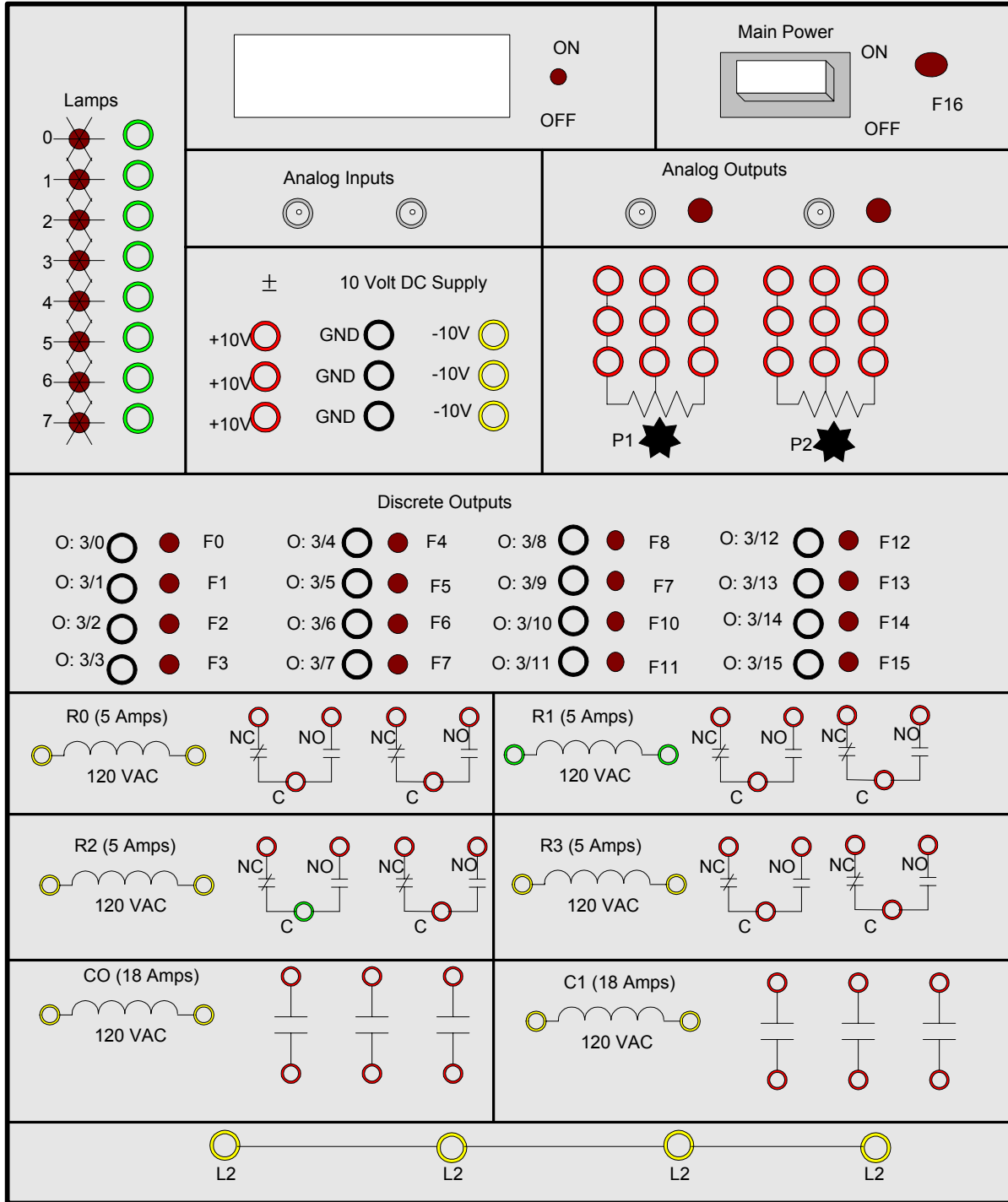


Figure 1.3 Bench Panel Output Devices

## 1.2 Ladder Diagrams

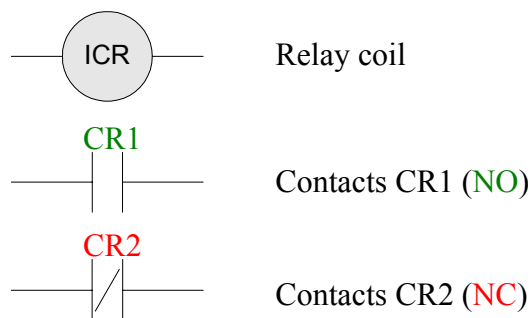
The logic implemented in PLCs is based on the three basic logic functions (AND, OR, NOT). These functions are used either singly or in combinations to form instructions that will determine if a device is to be switched ON or OFF. The most widely used languages for implementing ON/OFF control and sequencing are ladder diagrams. It is easy to understand, and most plant or industrial electricians are accustomed to working with elementary relay diagrams. Since this type of instruction set is composed of contact symbols, it is also referred to as contact symbology. The ladder circuit connections in the PLC are implemented via software instructions. All the logical wiring can be thought of as being inside the CPU (softwired as opposed to hardwired).

The complete ladder diagram can be thought of as being formed by individual circuits, each circuit having one output. Each of these circuits is known as a rung. Therefore, a rung is the contact symbology required to control an output in the PLC. A complete PLC ladder diagram program then consists of several rungs, each controlling an output interface which is connected to an output field device as shown in Figure 1.5. Each rung is a combination of input conditions (symbols) connected from left to right between two vertical lines, with the symbol that represents the output at the far right. The symbols that represent the inputs are connected in series, parallel or some combination to obtain the desired logic; these input symbols represent the input devices that are connected to the PLC's input interfaces. When activated, these devices either allow current to follow through the circuit or cause a break in current flow, thereby switching a device ON or OFF. The input symbols on a ladder rung can represent signals generated from connected input devices, connected output devices, or from outputs internal to the controller.

In general, PLC architecture is modular and flexible, allowing hardware and software elements to expand as the application requirements change. A PLC eliminates hardwired control in favor of programmable control. Once installed, the control plan can be manually or automatically altered to meet the day-to-day control requirements without changing the field wiring.

### 1.2 Contact Symbol

Programmable controller contacts and relay contacts operate in a very similar fashion. For example, the control relay CR shown in Fig. 1.4 has two sets of contacts, one normally open (CR1) and one normally closed (CR2).




### Figure 1.4 Relay Symbol with NO and NC contacts

If relay coil CR is not energized, contact CR1 will remain open and contact CR2 will remain closed. Conversely, if coil CR is energized, or turned ON, contact CR1 will close and contact CR2 will open. The following symbols are some of the basic bit instructions used to translate relay control logic to contact symbolic logic.

#### Symbol      Definition and symbol interpretation

 EXAMINE IF CLOSED INSTRUCTION:  
**XIC**

Generally referred to as normally open (NO) instruction. Typically represents any input to the control logic. The input can be a real world input from a connected switch or pushbutton, a contact from a connected output, or a contact from an internal output. The status bit is examined for an ON condition. If the status bit is "1", then the instruction is TRUE. If the status bit is "0", then the instruction is FALSE.

 EXAMINE IF OPEN INSTRUCTION:  
**XIO**

Generally referred to as normally closed (NC) instruction. Typically represents any input to the control logic. The input can be a connected switch or pushbutton etc., a contact from a connected output, or a contact from an internal output. The status bit is examined for an OFF condition. If the status bit is "0", then the instruction is TRUE. If the status bit is "1", then the instruction is FALSE.

 OUTPUT ENERGIZED:  
**OTE**

Typically represents any output that is controlled by some combination of input logic. An output can be a connected device or an internal output (internal relay). If the left-to-right path of input instructions in the rung is TRUE (all contacts closed), the output is energized (turned ON). If the rung is false, the output is reset to OFF.

 LATCH OUTPUT COIL  
**OTL**

An electromagnetic latching relay function can be programmed on a PLC to work like its real-world counterparts. This instruction is programmed if an output is to remain energized, even though the coil is energized only momentarily. The output is turned ON and remains ON until it is unlatched by an unlatch output instruction of the same reference address.

 UNLATCH OUTPUT COIL:  
**OTU**

This instruction is programmed to reset a latched output having the same reference output. If any rung path has logic continuity, the reference address coil is turned off

or unlatched to an off condition.

## 1.4 Address

Each symbol on the rung will have a reference number, which is the address in memory where the current status (1, 0) for the reference input is stored. When the field signal is connected to an input or an output interface, the address will also be related to the terminal where the signal wire is connected. The address for a given input/output can be used throughout the program as many times as required by the control logic.

The input addresses have the form I: e/b, where

- I = Input data file
- e = Slot number of the input module
- b = Terminal number used with input device

Similarly, output addresses have the form O: e/b

- O = Output data file
- e = Slot number of the output module
- b = Terminal number used with output device

It is important to remember that the program is not an electrical circuit but a logic circuit. In effect we are interested in logic continuity when establishing an output.

### Example 1

Figure 1.5 illustrates an example of a simple ladder program with the NO and NC contacts driving an output rung.

With the limit switch LS0 open, when SP0 is pushed the reference input SP0 (I: 1/0) is closed, the output coil PL1 (O: 3/0) is energized closing all (O: 3/0) contacts and the pilot light PL1 will be lit. Control contact (O: 3/0) in parallel with contact I: 1/0 then closes sealing in the output coil PL1 even if the start button is released. Since output PL1 (O: 3/0) is ON, the normally open contact (O: 3/0) in the second rung will close, turning the internal output coil INT\_OTTE\_1 (B3/1) ON. The NC contacts (O: 3/0) in the third rung will open because the examine for an OFF condition is not true (reference output is ON), therefore turning internal output INT\_OTTE\_2 (B3/2) OFF. Note that outputs B3/1 and B3/2 will not turn a real output device ON because these internal bits are not mapped to the output device. When the limit switch LS0 is closed the NC contact LS0 (I: 1/1) in the first rung opens, the output O: 3/0 is de-energized, and the pilot light PL1 is turned off. The normally open contact (O :3/0) in the second rung will open, turning the internal output coil INT\_OTTE\_1 (B3/1) OFF. The NC contact in the third rung is closed turning the internal output coil (INT\_OTTE\_2), B3/2 ON.



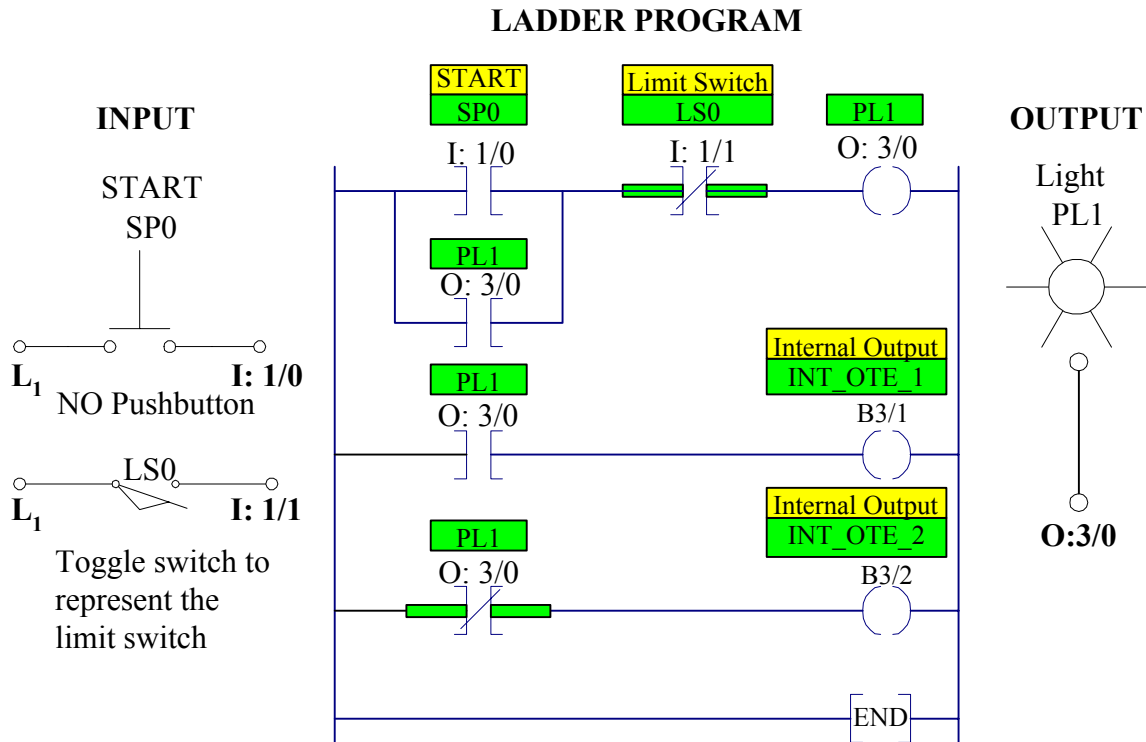


Figure 1.5 A simple ladder diagram with NO and NC contacts

## 2.1 Programming with RSLogix 500 Software

### Creating a directory on your network drive

To work correctly, the software needs a directory for your RSLogix programs on your f drive. Using either Windows Explorer or the “My Computer” icon on your desktop, create the directory **f:\RSLogix** on your network drive.

### Running RSLogix 500 and starting a new project

Double-click on the RSLogix 500 icon on the Windows™ desktop to start the software. With a new project, before you can edit a ladder program, you must configure the controller to reflect the actual hardware. The configuration procedure is outlined in section 2.2. Save this configuration in your F:\RSLogix directory with a file name say PLC500CONFIG.RSS. If you followed the configuration procedure the processor and I/O configuration has been completed, for your next project you can skip section 2.2 and copy the PLC500CONFIG file under a new file name, which you want to assign to your project.

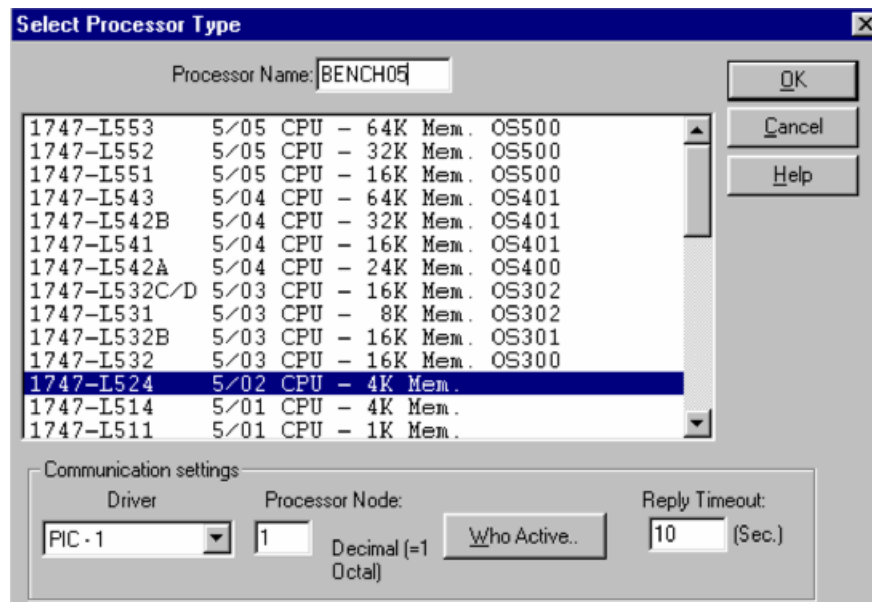
## 2.2 Processor and Input/Output Configurations

The PLCs in S-340 have a 10-slot backplane with the following modules:

Slot #	Part #	Description
0	1747-L524	5/02 CPU – 4K Mem.
1	1746-IA16	16-Input 100/120 VAC
2	1746-IA16	16-Input 100/120 VAC
3	1746-OA16	16-Output (Triac) 100/240 VAC
4	1746-OA16	16-Output (Triac) 100/240 VAC
8	1746-NIO4V	Analog 2 Channel In/2 Channel Voltage Out

You can go through the configuration process as follow:

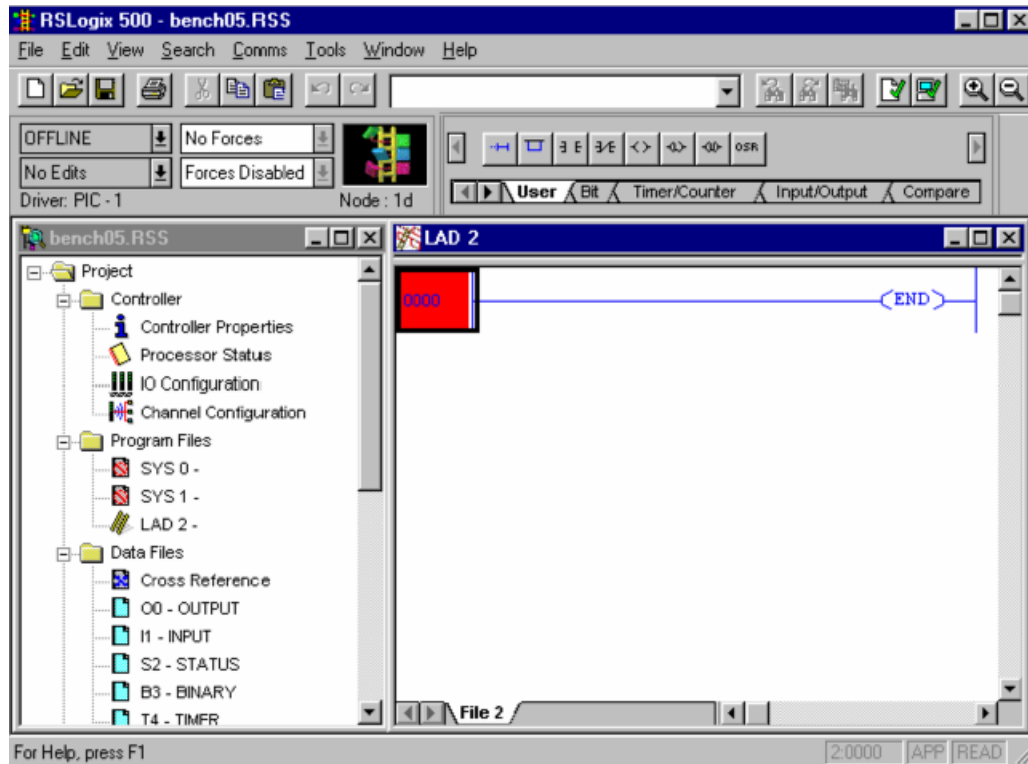
Double-click on the RSLogix 500 icon on the Windows™ desktop to start the software. To create a new project either click on the blank page icon or select “New...” from the “File” menu. A dialog box requesting processor information should appear, as shown



**Figure 1.6 Selecting Processor type.**

Select the **1747-L524 5/02 CPU** for the processor type and enter your bench number for the processor name.

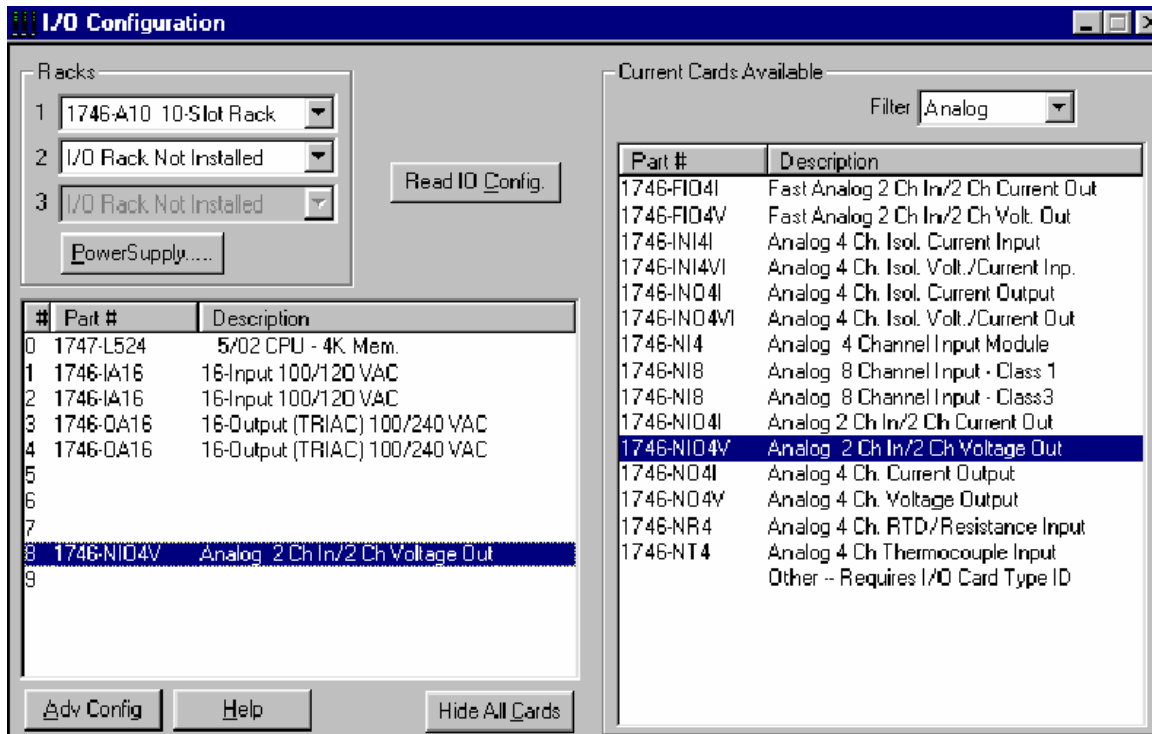
You should now see a project tree window and a window to enter your ladder logic.



**Figure 1.7 RSLogix 500 Project tree and Ladder Logic Windows.**

Next, the chassis (the rack in which the modules are installed) and the individual modules must be defined. Double-click on the **IO Configuration** icon in the project tree window to access the I/O Configuration dialog box. In the Rack 1 pull-down menu select 1746-A10 10-Slot Rack chassis. In slot 0 under Part # and Description you should see the 1747-L524 5/02 CPU – 4K Mem. Next, the I/O modules are defined. To do this, highlight slot 1 and double-click on the 1746-IA16 input module. Place the same input module in slot 2. Use slot 3 and 4 to configure the 1746-OA16 output modules. Finally place 1746-NIO4V analog module in slot 8.

A completed I/O Configuration dialog box is shown below.



**Figure 1.8 Completed I/O and Processor Configurations.**

Click on the X in the upper right corner to close the configuration dialog box.

You should now save the file by selecting “Save As” from the “File” menu. Save with a file name say PLC500CONFIG. We will use this file as a starting point (the base project file) for the remaining projects. For example, to start a new project you can copy the PLC500CONFIG file under a file name, which you want to assign to your project say example Lab1a.

### 2.3 Editing Ladder Logic

With RSLogix 500 Window in the OFFLINE mode, select the PLC500CONFIG file (i.e., the base project file that you configured in section 2.2) and save it under a new name, for example, Lab1a. You should see icons for the instructions you need for editing a program at the top of the RSLogix 500 programming window. A detailed description of the instruction appears in a balloon and on the status bar when you move the cursor over one of the instructions. Click on the tabs below the icons if you don't see an instruction you need. Use the following Step-by-step guide to edit the ladder diagram of Example 1 as shown in Figure 1.5.

1. Select the number of the first rung to begin entering instructions. The rung number should now be highlighted in red.
2. Click on the examine if closed (XIC) icon. An XIC instruction should now be on the rung.
3. Enter the instructions address. There are two ways to enter the instruction address:
  - a. Click on the question mark and type in the address I: 1/0.
  - b. Click on the I1-Input under project tree and drag the desired address (I: 1/0) to the instruction.

You may add a symbolic address. A symbol, once it has been associated with an address, can be used for other instructions referencing its address. To add symbols, click on the instruction and with address highlighted type over the address SP0 as the symbol for this instruction. You may also add a description for the instruction. To add a description, right click on the instruction and select “Edit Description.” A dialog box then appears for entering the description. You may type START for the description of this instruction.

4. Click on the examine if open (XIO) icon. An XIO instruction should appear on the rung next the XIC instruction. Follow step 3 for entering the address (I: 1/1), symbol (LS0), and description Limit Switch.
5. Click on the output enable (OTE) icon. An OTE instruction should appear at the end of the rung. For the addresses enter O: 3/0 and for the symbol type PL1.
6. Click on the branch icon. A branch appears with one end highlighted. Drag each end of the branch to its proper location. A green box appears at each possible location when you drag the branch ends.
7. Select the bottom left corner of the branch and click on the XIC icon. An XIC instruction is added to the branch. For its address you can type the symbolic address PL1 defined in step 5.
8. Select and double click on the End rung to insert a new rung. Add an examine if closed (XIC) instruction and type PL1 for its address. Next add the output enable (OTE) instruction. B3/1 (you may use B3 Binary icon under project tree and drag the desired address to the instruction). Type INT\_OTTE\_1 for the symbolic address, and Internal output for the description.
9. Repeat step 8 to add an XIO and an OTE instructions for the last rung.

After you have placed all the instructions, make certain they are all addressed. As a shortcut, you can drag addresses from one instruction to any other.

### Verify the program logic

Click on the file verify or project verify icon, or select them from Edit menu, to verify the program logic. If there are any logic errors in your program, a window appears below the project tree and ladder logic windows. Correct any errors before proceeding on with the lab.2.4

### Downloading Program To SLC-500

The *Upload/Download Program*, transfer programs between the computer and the SLC-500. By performing a download operation, you will be able to monitor your program when you enter the Online Programming mode. Turn on the processor Main Power and download the program of Example 1. The system displays a message stating:

Downloading Program

.....

Are you sure you want to proceed with Download?

If this is the correct file to download press *Yes*. If the processor was in the Run or Test mode

before you began downloading, the system asks if you want to switch the processor back to the PROG mode? Press *yes* to switch to the program mode. During the download procedure, the system displays a status screen showing the ladder and data table files being downloaded to the processor. After the program is downloaded, you can switch back to the desired mode.

## 2.5 Testing and Running the Program

Connect the Input/Output circuits shown in Figure 1.5. After the program has been successfully downloaded to the processor, use the following step to test and run the program.

### Testing mode

Select the **Test Continuous** mode. The system displays a message stating, Are you sure you want to change processor mode to Test? Press *Yes* to test the program.

In the test mode the ladder diagram executes, but outputs are not enabled.

To test the program press the start pushbutton SP0 the input address is forced, causing the output for that rung to be true and they will appear in reverse video. Control contact across pushbutton closes, sealing in the output coil PL1. On the second rung, since the condition is true, the NO contact and its output will appear in reverse video, and the internal bit B3/1 is turned on. On the third rung, the condition for an examine OFF is not true and the output for the rung is not energized and will not appear in reverse video.

Close the toggle switch LS0. On the first rung, the NC contact LS0 opens, and the output coil PL1 is turned off. On the second rung, the condition for an examine ON is not true and the output for the rung is de-energized and will not appear in the reverse video. On third rung, since the condition is true, the NC contact and its output will appear in reverse video, and the internal bit B3/2 is turned on.

### Running mode

Select the **Remote Run** mode. In the running mode the output is enabled, performing the intended operations. Press the start pushbutton and later close the toggle switch and observe the events made during test procedure.

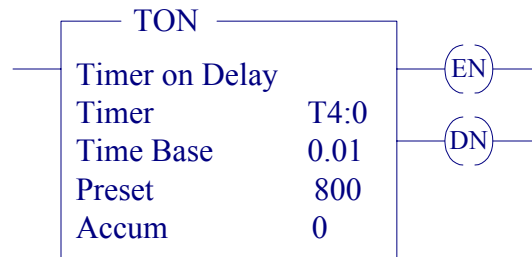
## 3. Timer and Counters

Timers and counters are internal instructions that provide the same functions as timing relays and counters. They are used to activate or de-activate a device after a preset interval of time. The timer is assigned an address as well as being identified as a timer. Also included, as part of the timer instruction is the *time base* of the timer, the timer's *preset value*, and the *accumulated value*. The timer instructions include:

### 3.1 Timer On-Delay (TON)

This instruction is programmed to provide time delay action. Once the rung has continuity, the

timer begins counting time-based intervals and times until the accumulated value equals the preset value. When the accumulated time equals the preset time, the output is energized, and the timed output contact associated with the output is closed. The timed contact can be used throughout the program as an NO or NC contact. The accumulated value is reset when rung condition goes false.



**Figure 1.9 Timer On-Delay Instructions**

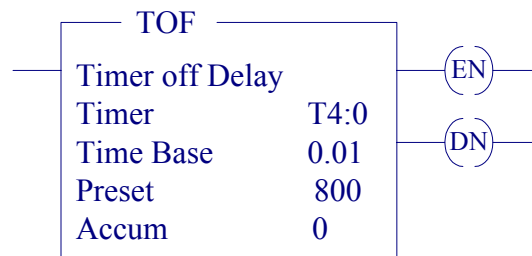
For SLC-500 processor the time base can be selected as 0.01 sec or 1.0 sec.

The control word uses three control bits:

- **Enable (EN) bit** The enable bit (EN) is set when rung conditions are true; it is reset when the rung condition becomes false.
- **Done (DN) bit** The done bit (DN) is set when the accumulated value is equal to the preset value. It is reset when rung condition becomes false.
- **Timer-timing (TT) bit** The Timer-timing bit (TT) is true when the timer is timing. When the timer is not timing the TT bit is false.

### 3.2 Timer Off-Delay (TOF)

This instruction is programmed to provide time delay action. If the rung does not have continuity, the timer begins counting time-based intervals and times until the accumulated value equals the preset value. When the accumulated time equals the preset time, the output is energized, and the timed output contact associated with the output is closed. The timed contact can be used throughout the program as an NO or NC contact. The accumulated value is reset when rung condition goes true.



**Figure 1.10 Timer Off-Delay Instruction**

The done bit (DN) is set when the accumulated value is equal to the preset value. It is reset

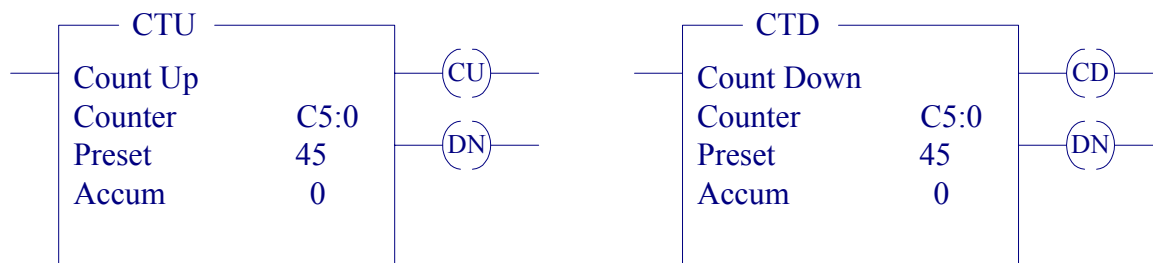
when rung condition becomes true. The enable bit (EN) is set when rung conditions are true; it is reset when the rung condition becomes false.

### 3.3 Counters

Counters are similar to timers, except that they do not operate on an internal clock, but are dependent upon external or program sources for counting. The counter is assigned an address as well as being identified as a counter. Also included, as part of the counter instruction is the counter's *preset value* as well as the current *accumulated count* for the counter. There are two basic types of counters, one that counts up and another one that can count down.

#### Count Up (CTU) and Count Down (CTD)

The up-counter output instruction will increment by one and the down-counter output instruction will decrement by one each time the counted event occurs. These events could be caused by the number parts traveling past a detector or a limit switch. When the accumulated counts equal the preset count, the output is energized, and the counter output is closed. The counter contact can be used throughout the program as an NO or NC contact



**Figure 1.11 Count Down and Count Up Instruction**

The done bit (DN) is set and remains set when the accumulated value is equal to the preset value. It is reset when rung condition becomes true. The count up enable bit (CU) or the count down enable is set when rung conditions are true; it is reset when the rung condition becomes false or the appropriate reset instruction is enabled.

#### Counter and Timer Reset (RES)

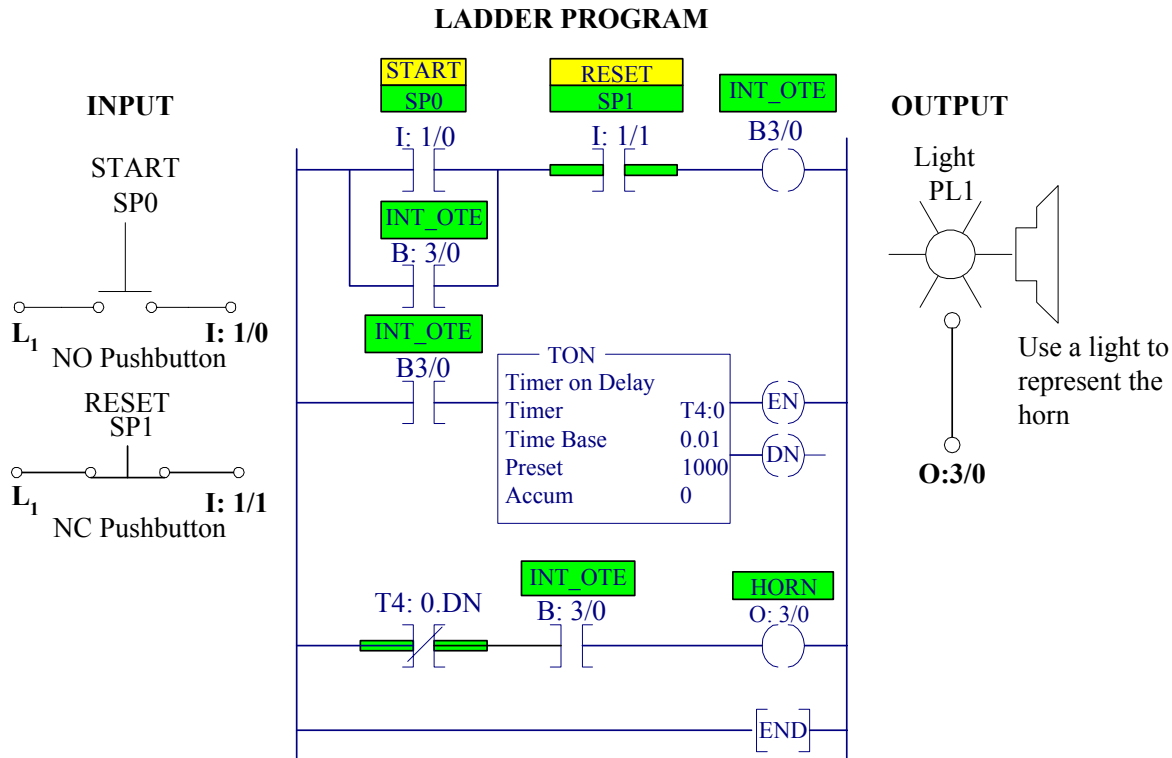
The [RES] instruction is used to reset the timing and counting instruction accumulated values. A rung containing an NO or an NC contact together with the [RES] instruction is used. The [RES] instruction must be given the same reference address as the related timer or counter. When the [RES] instruction is enabled, the counter accumulated value is reset.

#### Example 2

Figure 1.12, a logic ladder program, which can be used as a warning signal when moving equipment, such as a conveyor motor, is about to be started. The internal output INT\_OTE is energized when the START pushbutton SP0 is momentarily actuated. As a result contact B3/0 across SP0 closes to seal in the output coil INT\_OTE. Also, contact B3/0 on the second rung closes to energize timer coil, and contact B3/0 on the third rung closes to sound the horn. After a



10 second delay period, the NC timer contact (4:0.DN) opens to automatically switch the horn off. Pressing the REST Pushbutton SP1 will reset to the initial condition.



**Figure 1.12 Starting-up warning signal circuit**

Refer to section 2.3 to define a new file name and edit Figure 1.12 ladder diagram.

### Test and Run Example 2

Connect the Input/Output circuits shown in Figure 1.12. Follow the procedure in parts 2.4 and 2.5 to download Example 2 to SLC-500, to test and run the project.

Press the start pushbutton SP0, the input address is forced, causing the output for that rung to be true and they will appear in reverse video. Control contact across pushbutton SP0 closes, sealing in the internal output INT\_OTE. On the third rung, since the condition is true, the output at O: 3/0 is turned on and appears in reverse video. On the second rung, since the condition is true, the timer is energized. After 10 seconds timer is enabled opening the timer contact (T4: 0.DN) in the third rung. This will de-energize output coil and turn off the horn. The timer-accumulated value is reset to zero.

## 4. RSLogix 500 Program Reporting

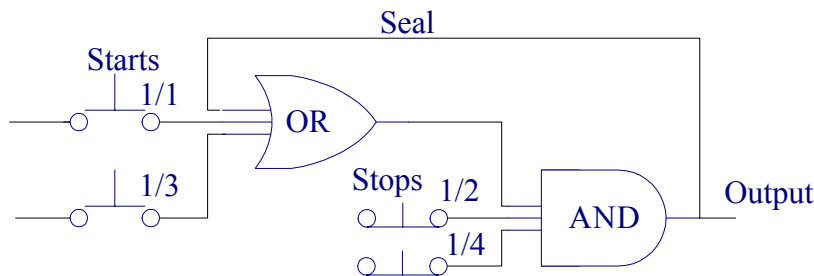
To send a report of your project to the file server HP LaserJet, from the SLC-500 Main Menu, you can select which reports you want to print by accessing File > Report Options. Select a check box on the Report Options dialog to print that report. If you'd like a step-by-step procedure for selecting and printing a report, from the file menu select Report Options and refer to the help menu.

### Pre-Lab for next session

Edit the ladder logic program for the following two exercises. For Exercise 2 study the timer operation and the ladder logic program for project 2 (Motor Starter Control) given in the LABORATORY SESSION 2. Your work will be checked in the next session.

### Exercise 1

A motor control circuit has two start and two stop buttons. When either of the start buttons is depressed, the motor runs. It is required that the start buttons be latched so that, after it has been pressed, the output remains on until either stop buttons are pressed. Figure 1.13 shows the gate logic for this task. Use RSLogix 500 software to edit a ladder logic program. Note that the AND gate is converted to series contacts and the OR gate is converted to parallel contacts.



**Figure 1.13 Gate Logic for Exercise 1**

Connect the Input/Output circuits for this exercise. Download the ladder logic to SLC-500, test and run Exercise 1.

### Exercise 2

Construct a ladder logic program for a washing machine to switch a pump for 30 second. It is then to be switched off and a heater switched on for 20 seconds. Then the heater is switched off and another pump is started for 30 seconds to empty the water. (Hint: use the DN and TT control bits).

Connect the Input/Output circuits for this exercise. Download the ladder logic to SLC-500, test and run Exercise 2.

### 5. Report Requirement

Include a printout of the ladder logic programs for examples 1 and 2, and exercises 1 and 2.

Describe each control circuit. Outline the step-by-step sequence of operation and discuss the observations made during the operation for each example in this Lab.