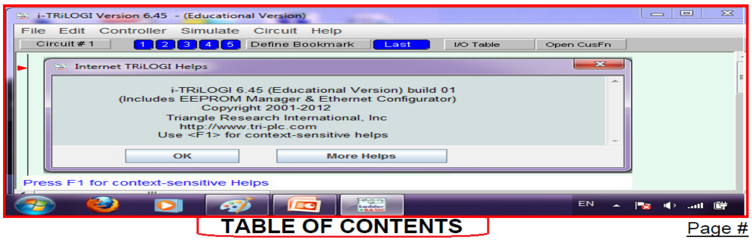
PLC Simulator Sofware: i-TRiLOGI Ladder+BASIC Version 6.45

Client/Server type program. 1st part (called Client program) is editor, compiler, simulator&program up-loader software



i-TRiLOGI 6.45 (Educational Version) build 01
(Includes EEPROM Manager & Ethernet Configurator)
Copyright 2001-2012
Triangle Research International, Inc
http://www.tri-plc.com
Use <F1> for context-sensitive helps

Press F1 for context-sensitive Helps

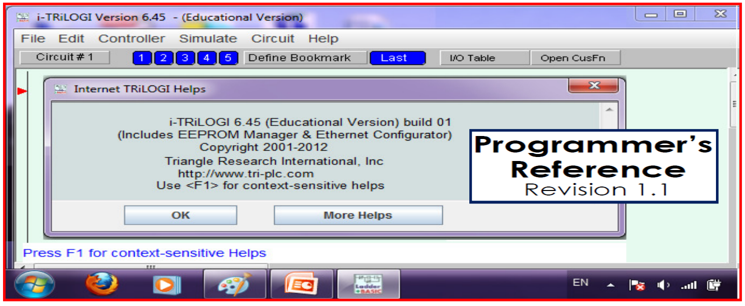
TABLE OF CONTENTS

Page #

PLC Simulator Sofware: i-TRiLOGI Ladder+BASIC Version 6.45

Client/Server type program. 1st part (called Client program) is editor, compiler, simulator&program up-loader software

البرنامج الأول لتعليم البرمجة والثانى غيرموجود (TLServer 3.x) للتخاطب مع server لرفع برنامج LAD على النت ثم يمكن تنزله على PLC مباشرة
بعد الإطلاع على كافة قائمة محتويات البرنامج للتعرف على أهم ما يقوم به من وظائف ، **سوف يكون تركيزنا فقط على الفصل السادس بصفة خاصة** لأنه
موضوع التقرير (المطلوب عن البرنامج كجزء من أعمال الترم) وباقى الفصول من السابع للعاشريمكن كذلك دراستها بصفة عامة للمساعدة فى كتابة التقرير.



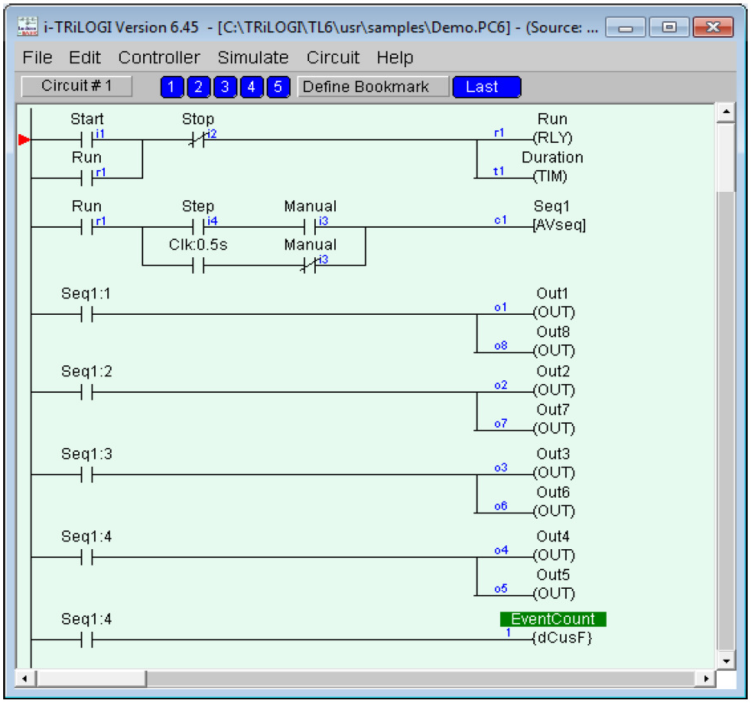
i-TRiLOGI 6.45 (Educational Version) build 01
(Includes EEPROM Manager & Ethernet Configurator)
Copyright 2001-2012
Triangle Research International, Inc
http://www.tri-plc.com
Use <F1> for context-sensitive helps

Programmer's Reference
Revision 1.1

Press F1 for context-sensitive Helps

Your Assignment: Creating Your First Ladder Logic Program

In this tutorial, we would like to create a simple program as shown below:



Simply follow the steps below to create your first ladder logic circuit.

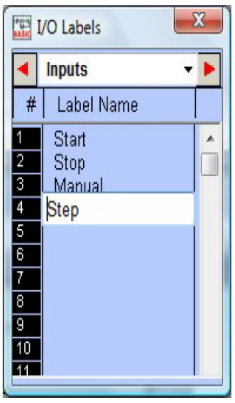
- Open pull-down "File" menu and select "New".
- You should now be in the "Browse" mode of the logic editor. The vertical line on the left end of the screen is the "power" line. The cursor is at the position where you can key in your very first ladder logic.

You can also view a video version of this tutorial here.

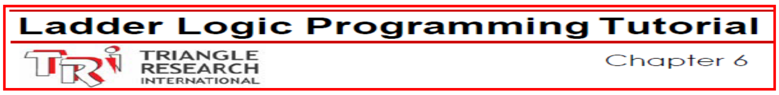
6.1 Ladder Logic Programming Tutorial: STEP1

Before we commence the circuit creation, let us define the I/Os to be used for this program. The following I/Os are required:

	Name
Inputs :	Start, Stop, Manual, Step
Outputs :	Out1, Out2,.... Out8
Relays :	Run
Timers :	Duration
Sequencer (counter) :	Seq1



1. Click on the I/O Table button located on right hand side of the upper status bar to open up the I/O label editing Window. You can also achieve the same with the <F2> function key.



Ladder Logic Programming Tutorial

TRIANGLE RESEARCH INTERNATIONAL

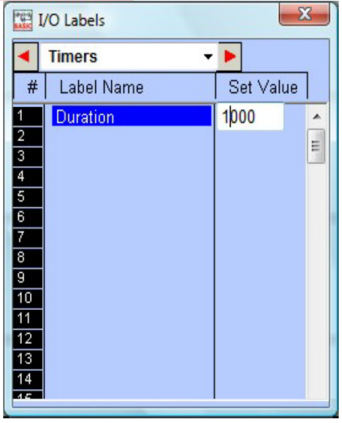
Chapter 6

2. Scroll to the "Inputs" window by using the left/right cursor keys or by clicking on the red color left/right arrow buttons or simply select it from the choice box between the left/right arrow buttons.
3. Move the deep blue color highlight bar to Input #1 position by clicking on it. Click again to open up a text field for entering the name for Input #1.
4. Press <Enter> key again and the highlight bar will be moved to Input #2.
5. Without using the mouse button, simply start typing the name "Stop" at Input #2. The text field will be automatically opened up at Input #2 for entry. Press <Enter> after typing in the name for "Stop" input.
6. Complete entry of the other two input label names "Manual" and "Step" as above. Note that i-TRiLOGI Version 6.2 and up allows I/O label names of up to 16 characters. However, if you wish to keep compatibility with Version 5.x to 6.1x, you should use no more than 10 characters to define the I/O names. Also, white spaces between names are not acceptable and will be automatically converted to the underscore character ('_'). e.g. If you enter the name: "F series PLC" for an I/O, it will be accepted as "F_series_P".
7. After entering label names for Inputs #1 to #4, move to the "Output" table by pressing the right cursor key or by clicking on the right arrow button. Enter all the output and relay label names in their respective I/O tables. We will discuss the "Timer" table in the next step.

IMPORTANT

1. You can shift the Items in the I/O table up or down or insert a new label between two adjacent, pre-defined labels. Simply press the <Ins> key or Right-Click the mouse button to pop up the "Shift I/O" menu, which allows you to shift the selected I/O. However, please note that if you shift the I/O down, the last entry in the I/O table (e.g. Input #256) will be lost.
2. In i-Trilogi 6.2 and up, shifting of Custom Function Label names will now shift the function content along with the label name. (In previous versions of i-Trilogi, shifting of the I/O label would not shift the function content, therefore making it untenable to use I/O label shift to reorganize custom functions. Warnings are provided if such an action were to result in overwriting of an existing custom function.
3. i-TRiLOGI Version 6.2 and up allows I/O label names of up to 16 characters. However, if you wish to keep compatibility with Version 5.x to 6.1x, you should use no more than 10 characters to define the I/O names.

6.2 Ladder Logic Programming Tutorial: STEP 2



Label Name: Duration
Set Value: 1000

1. Timer table has an extra column "Set Value" located to the right of the "Label Name" column.
2. After you have entered the label name "Duration" for Timer #1, a text entry box is opened up at the "Set Value" location of Timer #1 for you to enter the SV for the timer. SV range is between 0 and 9999. Enter the value 1000 at this location.
3. For a normal timer with 0.1s time base, the value 1000 represents 100.0 seconds, which means that the "Duration" timer will time-out after 100.0 seconds. If the timer had been configured as "High Speed Timer" using the TBASIC "HSTimer" command, then the time-base would become 0.01s, meaning the value 1000 represents only 10.00 seconds.
4. We are now left to define the sequencer, "Seq1". The sequencer is an extremely useful device for implementing sequencing logic found in many automated equipment. i-TRiLOGI supports 8 sequencers of 32 steps each. Each sequencer requires a "Step counter" to keep track of the current step sequence.

   The first 8 counters in the counter table double as the step counters for the 8 sequencers. These sequencers must be named "Seq1" to "Seq8" if they are to be used, i.e. Counter #1 to be named as "Seq1", Counter #2 as "Seq2", etc. However, any counter not used as sequencer may assume any other name (up to a maximum of 10 characters) if they are used as ordinary counters.

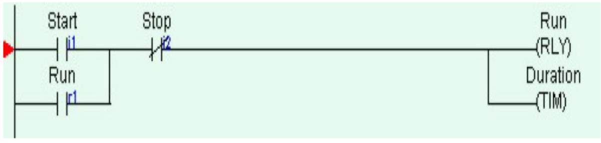
   If you are at the "Timers" table, pressing the right cursor key again will bring up the "Counters" table. Enter the name: "Seq1" at the label column for Counter #1. Press <Enter> and the text entry field will be opened at the "Set Value" column. For now, let's enter a preset value of "4" for "Seq1".

يوجد فى help وفى نهاية هذا الفصل جزء شرح مفصل للفرق بين Sequencer&Counter

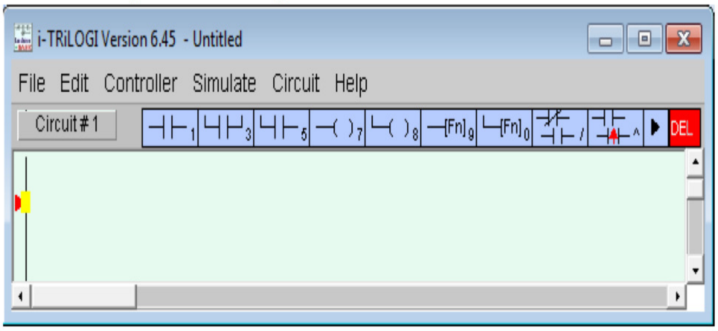
5. We have now completed defining the I/Os, timers and counters. Press the <ESC> key to close the counter or other tables. Note that not all labels need to be defined before programming. You may create the label names any timer during circuit creation by pressing hotkeys <F2>.

6.3 Ladder Logic Programming Tutorial: STEP 3

We are ready to create Circuit #1 as shown below:



1. With the circuit pointer (red color triangle) at Circuit #1, press the <Spacebar> to enter the "Ladder Edit" mode. You can also enter the circuit edit mode by double clicking at Circuit #1.

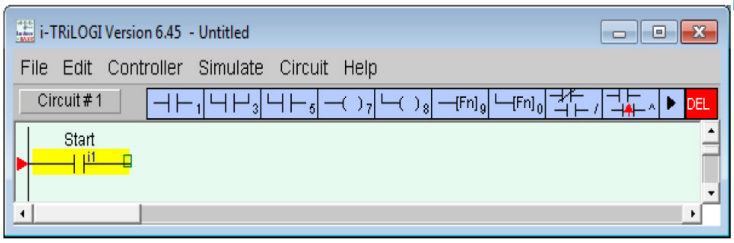


Once you enter the "Ladder Edit" mode, a row of ladder icons appear along the top of the main i-TRiLOGI window just below the pull down menu. The following is a description of each item. A yellow color highlight bar, which you can move to select an element in the ladder circuit, will appear.

⊣⊢1	<1> - Left click to insert a normally-open series contact. <2> - Right click to insert a normally-closed series contact.
⊣⊢3	<3> - Left click to insert a N.O. parallel contact to highlighted element <4> - Right click to insert a N.C. parallel contact to highlighted element
⊣⊢5	<5> - Left click to insert a N.O. parallel contact to enclose one or more elements. <6> - Right click to insert a N.C. parallel contact to enclose one or more elements.
–( )7	<7> - Insert a normal coil which may be an output, relay, timer or counter.
( )8	<8> - Insert a parallel output coil (not an entire branch) to the current coil.
–{Fn}9	<9> - Insert a special function coil which includes execution of CusFn
{Fn}0	<0> - Insert a parallel special function coil to the current coil.
⊣/⊢ /	</> - Invert the element from N.O. to N.C. or from N.C. to N.O.
⊣⊢ ^	<^> - Convert the element to a rising-edge triggered contact (one shot)
▶	Click to move the highlight bar to the right (same effect as pressing the right arrow key). This can be used to move the cursor to a junction which cannot be selected by mouse click.
DEL	Double-click to delete a highlighted element. This acts as a safety against mistake.

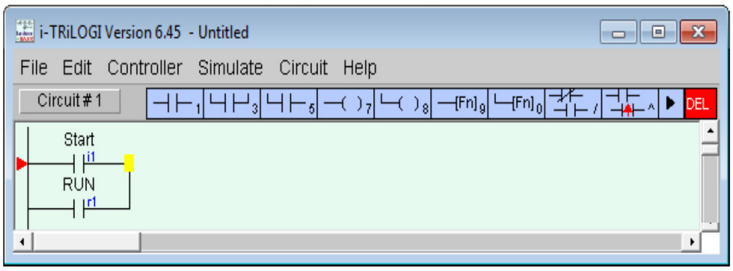
2. Now insert the first element by left-clicking on the ⊣⊢1 icon. The icon will change to a bright yellow color to show you the element type that you are creating. At the same time, an I/O table should appear on the screen with a light beige-color background instead of the normal light blue background. The I/O table now acts like a pop-up menu for you to pick any of the pre-defined label names for this contact.

   Note: In i-TRiLOGI version 6.x, if you pick any undefined I/O you will be prompted to enter the label name and what you entered will automatically be updated in the I/O table.
3. The contents in the table are not normally meant to be edited at this

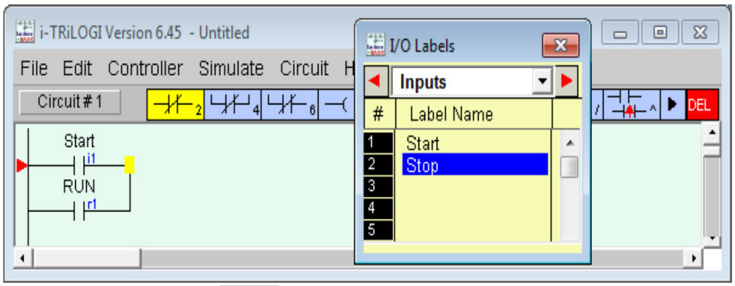


If you observe the highlight bar carefully, you will notice a dark green color square at the right end of the highlight. This indicates the insertion location where a series contact will be attached. You can change the insertion location to the left or the right of the highlight bar by pressing the <SHIFT> key or <TAB> key (Note: <TAB> key only works on JRE 1.3.1 or earlier. It does not work on JRE 1.4.x)

4. Next, create the contact "RUN" which is parallel to the "Start" contact by left-clicking on the ⊣⊢3 icon. The I/O table will appear again. Scroll to the "Relay" table and select the "RUN" relay.
5. To insert the normally-closed "Stop" contact in series with the "Start" and "Run" contacts, you need to move the highlight bar to the junction of the "Start" and "Run" contact. First click on the "Start" contact to select it. Then click on the ▶ icon to move the highlight bar to the junction, as follow:

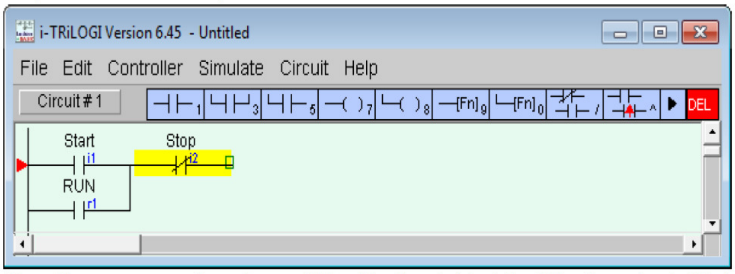


Note: The highlight bar will move to the junction if you click on the dark green insertion point on the "Start" contact.


i-TRiLOGI Version 6.45 - Untitled
File Edit Controller Simulate Circuit H
Circuit # 1
Start
RUN
DEL
I/O Labels
Inputs
Label Name

#	Label Name
1	Start
2	Stop
3	
4	
5	

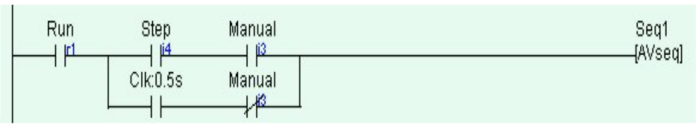
6. Next, right-click on the ⊣⊢ icon. It will change into yellow color normally-closed contact as shown in the above diagram. You are now inserting a normally-closed series contact at the location of the highlight bar. Pick the "Stop" label from the "Input" table to add the series contact.


i-TRiLOGI Version 6.45 - Untitled
File Edit Controller Simulate Circuit Help
Circuit # 1
Start
Stop
RUN
DEL

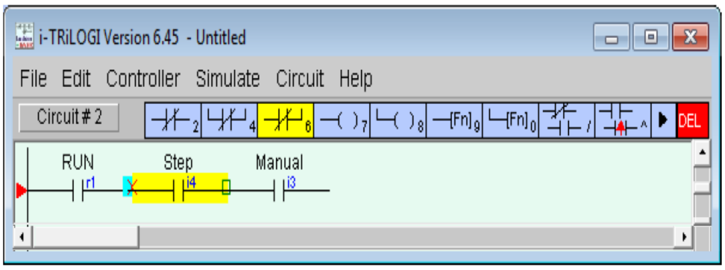
7. We will now connect a relay coil "Run" to the right of the "Stop" contact. Click on the –( ) icon to insert the coil. Select "RUN" label from the "Relay" table. Remember that an input can never be used as a coil. Fortunately, i-TRiLOGI is smart enough not to call up the "Inputs" table when you are connecting a coil, to avoid unintentional errors.
   Notice that the coil symbol ---(RLY) indicates that this is a relay coil, which is helpful in identifying the function of the coil. i-TRiLOGI automatically places the coil at the extreme right end of the screen and completes the connection with an extended wire.
8. Right below the relay coil is a parallel timer coil with label name "Duration". To create this coil, click on the └( ) icon. This allows you to connect a parallel coil to the existing coil. The "I/O" table will pop up for selection again. Since we want to choose a timer, scroll to the "Timer" table and pick the first timer with the label "Duration" to complete the circuit.
   Press the <Enter> key once to complete Circuit #1
9. Congratulations! You have just successfully created you very own ladder logic circuit. It is that simple! It may be a good time to save your program now by pressing <CTRL-S> key or select "Save" from the "File" menu and give a file name for you new program.

6.4 Ladder Logic Programming Tutorial: STEP 4

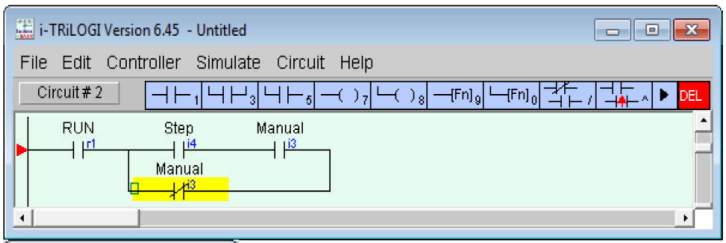
We will now create Circuit #2 as shown below.

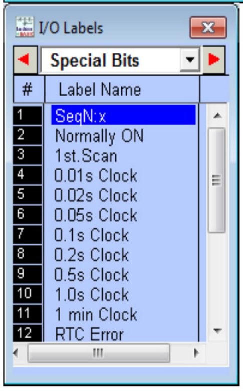


1. Follow the steps listed in STEP 3 to create the following circuit fragment:


i-TRiLOGI Version 6.45 - Untitled
File Edit Controller Simulate Circuit Help
Circuit # 2
RUN
Step
Manual
DEL

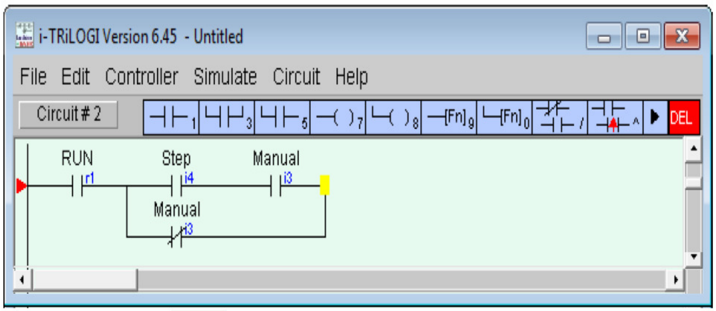
2. We want to enclose the two series contacts "Step" and Manual" with a parallel branch that contains two elements. First, we will create the branch for the N.C. "Manual" contact.
3. Click on the element "Step" to highlight it. Then right-click on the ⊣⊢ icon to create a N.C. parallel circuit that encloses both the "Step" and the "Manual" contacts. A cross will appear at the left hand end of the "Step" contact, indicating that this is the starting location of the parallel circuit. You should now click on the "Manual" contact to select the ending location for the parallel circuit. The yellow highlight bar will be positioned at "Manual" contact now.
4. You will notice that the ⊣⊢ icon has now changed into a yellow color N.C. contact ⊣/⊢ with an opposite connection arm. You should now click on the ⊣/⊢ symbol to close the parallel branch. (One possible short-cut method is to double-click at the ending location to close the branch).
   As usual an I/O table will be opened for you to select the I/O. For now, select the "Manual" label from the "input" table to create the following circuit:


i-TRiLOGI Version 6.45 - Untitled
File Edit Controller Simulate Circuit Help
Circuit # 2
RUN
Step
Manual
Manual
DEL

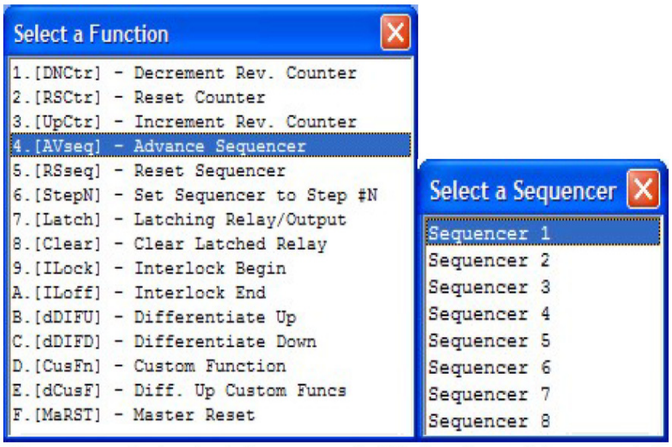


#	Label Name
1	SeqN:x
2	Normally ON
3	1st.Scan
4	0.01s Clock
5	0.02s Clock
6	0.05s Clock
7	0.1s Clock
8	0.2s Clock
9	0.5s Clock
10	1.0s Clock
11	1 min Clock
12	RTC Error

5. Next, we want to insert the special bit "Clk:0.5s" to the left of the "Manual" contact. Press the <SHIFT> key to move the insertion point to the left end of the highlight bar as shown above. Then left-click on the ⊣⊢ icon to create a normally-open contact. Scroll the I/O table to the "Special Bits" table and select the item: "0.5s Clock". The parallel branch would have been completed by now.
   Note: The "Special Bit" table comprises some clock pulses and some other special purpose bits. These include the eight built-in clock pulses in the system with periods ranging from 0.01s to 1 minute. Built-in clock pulses are useful if you need a time base to create, for example, a "flashing light". A contact such as "Clk:0.1s" will automatically turn itself ON for 0.05s and then OFF for another 0.05s and then ON again, resulting in a series of clock pulses of period = 0.1 second.
6. Next, move the highlight bar to the right end junction of the parallel circuits as follow:


i-TRiLOGI Version 6.45 - Untitled
File Edit Controller Simulate Circuit Help
Circuit # 2
RUN
Step
Manual
Manual
DEL

7. Now, click on the –[Fn] icon to insert a special function coil. A popup menu will appear for you to select the desired special function. Click on the item "4.[AVseq]-Advance Sequencer" to insert the Advance Sequencer function [AVseq].



Select a Function

1.[DNCtr] - Decrement Rev. Counter
2.[RSCtr] - Reset Counter
3.[UpCtr] - Increment Rev. Counter
4.[AVseq] - Advance Sequencer
5.[RSseq] - Reset Sequencer
6.[StepN] - Set Sequencer to Step #N
7.[Latch] - Latching Relay/Output
8.[Clear] - Clear Latched Relay
9.[ILock] - Interlock Begin
A.[ILoff] - Interlock End
B.[dDIFU] - Differentiate Up
C.[dDIFD] - Differentiate Down
D.[CusFn] - Custom Function
E.[dCusF] - Diff. Up Custom Funcs
F.[MaRST] - Master Reset

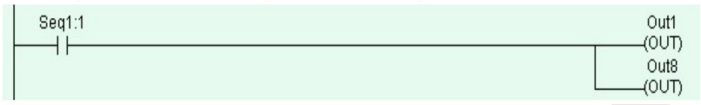
Select a Sequencer

Sequencer 1
Sequencer 2
Sequencer 3
Sequencer 4
Sequencer 5
Sequencer 6
Sequencer 7
Sequencer 8

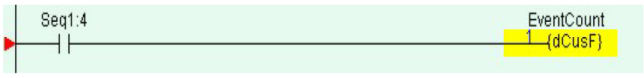
8. When prompted, select Sequencer 1. This function will increment the step counter of Sequencer #1 each time its execution condition goes from OFF to ON. Again, remember to press the <Enter> key to complete Circuit #2

6.5 Ladder Logic Programming Tutorial: STEP 5

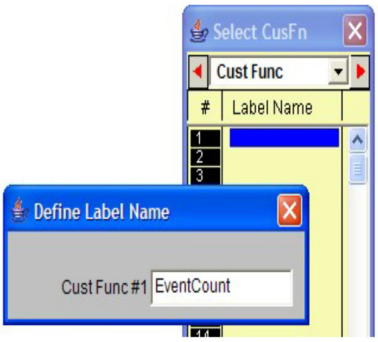
1. Circuits #3 to #6 are similar to one another. They make use of the Sequencer to turn on the Outputs 1 to 8 to create a pattern of "running lights" when executed. The label "Seq1:1" of the contact in Circuit #3 represents Step #1 of Sequencer 1. Remember that each sequencer can have up to 32 steps (Step #0 to 31), with each step individually accessible as a contact. A normally-open contact "Seq1:1" will be closed whenever the step counter of Sequencer 1 reaches number 1. Likewise a normally-closed contact "Seq5:20" will be opened when the step counter of Sequencer 5 reaches number 20.



2. To create the normally-open contact "Seq1:1", left-click on the ⊣⊢ icon. When the I/O table pops up, scroll to the "Special Bit" table and select the item #1 "SeqN:x". When prompted to select a sequencer choose "Sequencer 1" and another dialog box will open up for you to enter the specific step number for this sequencer. At this point, you should enter the number "1" since we are using Step #1.
3. We have thus far been creating ladder circuits only by clicking on the ladder icons. A short-cut method of choosing elements to be created without using the mouse does exist. Observe the icon carefully and you will notice a small numeral at the lower right hand corner of each icon which correspond to the shortcut key. You may wish to try this short-cut for the remaining part of Circuit #3. Press the <7> key and the Output table will immediately pop up for selection of a coil. Pick "Out1" from the "Output" table and the "Out1" coil will be connected.
4. Circuits #4, 5 and 6 are very similar to Circuit #3 and you shouldn't have problem creating them. Complete these circuits and we are ready for some interesting simulation exercises. When you have created all the circuits, press <Enter> key or <ESC> key at the last blank circuit to end "Ladder Edit" mode.
5. Next, we want Seq1:4 contact to execute a "Custom Function" (the abbreviation is CusF). A custom function is one that you can create to perform very sophiscated control actions using the TBASIC language. The Custom function can contains multiple TBASIC statements that gives the PLC extremely powerful capabilities unmatched by many ladder-only type of PLCs. The Circuit we wish to create is as follow:



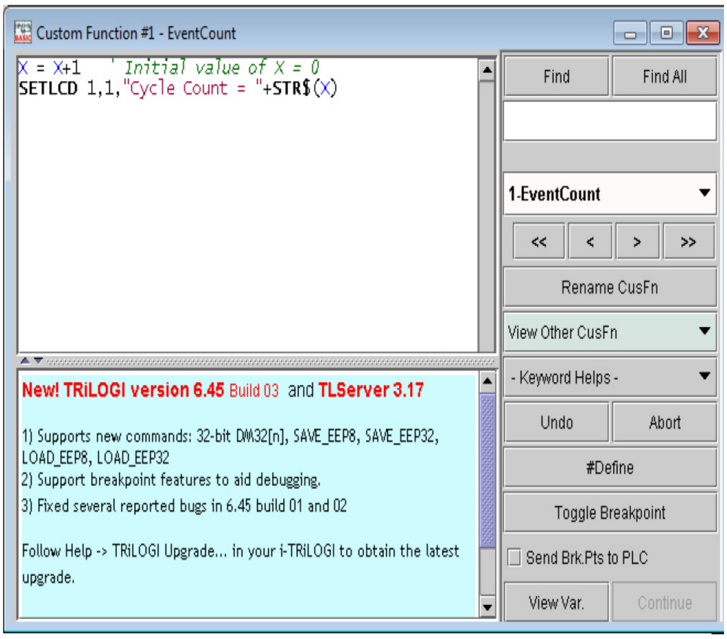
6. First, create the contact "Seq1:4" as per previous circuits. Then left-click on the —{Fn} to connect to a Special Function coil. Select from the popup menu the item "E: Diff. Up Custom Function" to create a "Differentiated Up" version of Custom Function (For a full explanation of the meaning of Differentiated Up", please refer to TBASIC Introduction). The Custom function popup menu will appear for you to select the Custom Function number you wish to insert. Up to 256 Custom functions can be defined and you can click on the first custom function, as follows:



Cust Func #1 EventCount

7. Since the custom function has not been defined with a label name, you will be prompted to give it a label name. The label name given to the CusF must be unique and are subject to the same limitations as that of the I/O label names. Lets enter the label name "EventCount" for function #1.

   Note: Label name for Custom function is optional and if you don't wish to use label name then the program will automatically assign it a name "Fn_#1" for function #1, "Fn_#2" for function #2, and so on.
8. When the Custom Function editor has been opened, enter the following text and then close the editor window by click on the ☒ or by pressing the <ESC> key. If you would like to exit the editor window without saving the changes, click on the Abort button.



```
X = X+1    ' Initial value of X = 0
SETLCD 1,1,"Cycle Count = "+STR$(X)
```

New! TRiLOGI version 6.45 Build 03 and **TLServer 3.17**

1) Supports new commands: 32-bit DM32[n], SAVE_EEP8, SAVE_EEP32, LOAD_EEP8, LOAD_EEP32
2) Support breakpoint features to aid debugging.
3) Fixed several reported bugs in 6.45 build 01 and 02

Follow Help -> TRiLOGI Upgrade... in your i-TRiLOGI to obtain the latest upgrade.

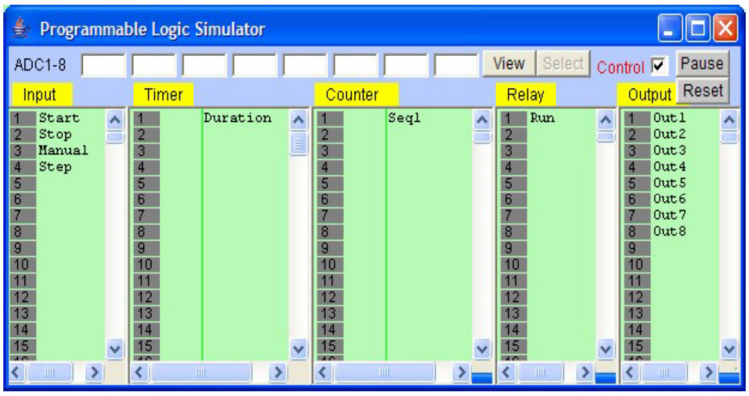
The purpose of this custom function is to keep a cycle count of the number of times the Sequencer has completed the complete sequencing and display the cycle count on the PLC's built-in LCD display (or on the Simulator's "View Variable" screen). This custom function should be run ONLY ONCE whenever the Seq1:4 contact closes, which is why the differentiated version of Custom function must be used in this case ({dCusF}, not {CusFn})

9. We can make our program more comprehensive to other users by utilizing the "Comments" feature of i-TRiLOGI. Open the "Circuit" menu and select "Insert Comment". A comment editor window will be opened up to allow you to add your comments to any part of the circuit. When you are done with your comments, just press <ESC> key or close the comment editor window and the comments you just entered will be inserted between the circuits. Each comment occupies a circuit position and there is no limit to the number of lines a comment circuit may have. (However, if you wish to keep data file compatibility with the old DOS TRiLOGI Version 4.x you should limit the comment to no more than 4 lines per comment and each line should contain no more than 70 characters.)

   A comment circuit may be moved around or deleted just like any other ladder circuits. If you wish to edit the comment, just double-click on it or press the <Spacebar> to open up the comment editor window. You can use the normal text editing keys such as left, right, up, down cursor keys, and <Ctrl-Left>, <Ctrl-Right>, <Del> and <Backspace> keys for editing comment text.

6.6 Ladder Logic Programming Tutorial: STEP 6

The stage has been set and the show is ready! Having completed the demo program, it is time to test if it works as intended using the built-in real-time programmable controller simulation engine. Open the "Simulate" pull-down menu and activate the command "Run (All I/O reset) - Ctrl+F9". i-TRiLOGI will immediately compile the ladder program and if no error is detected, it will instantly proceed to open up the "Programmable Logic Simulator" screen, as shown below:

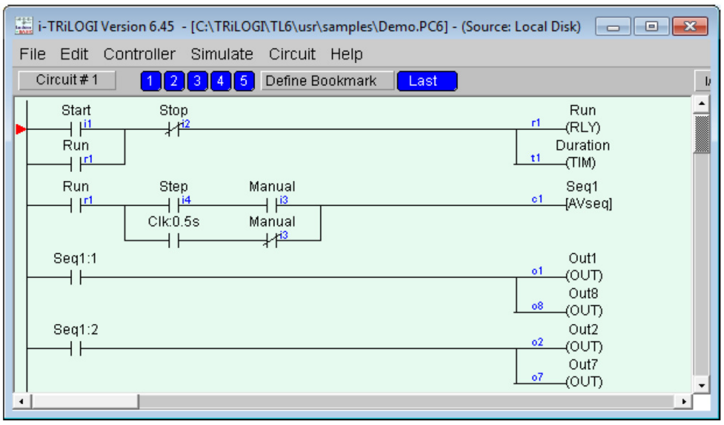


1. If you have followed closely all the instructions during the creation of the demo program, you should not encounter any compilation error. However, if you do receive an error message, then please check your circuit against the picture shown in the assignment page, make all necessary corrections and then try again.
2. The simulator screen comprises 5 columns: Input, Timer, Counter/Sequencer, Relay, and Output. With the exception of the Relay table which contains up to 512 elements, and the Timer table which contains up to 128 timers, all other columns contain 256 elements each. Every column has its own vertical scroll bar. You can use the mouse to scroll each column independently to locate the desired I/O.
3. The label names for the inputs, outputs, relays, timers and counters defined earlier in the I/O tables automatically appear in their respective columns. To the left of each label name column is an "LED" lamp column which indicates the ON/OFF state of respective I/O. A red color lamp represents the ON state of an I/O, whereas a dark grey color lamp represents an OFF state. The I/O number is indicated in the middle of the lamp.

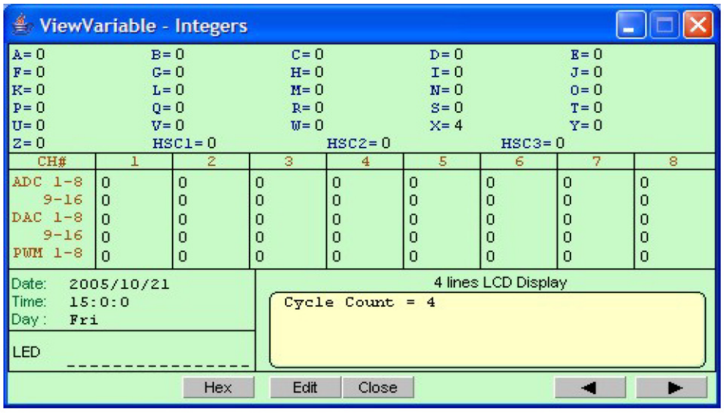
   The simulator requires the use of the mouse to work properly, so it is important to remember the mouse button actions as follows:

Left Mouse Button	Turn ON the I/O when pressed. Turn OFF when button is released.
Right Mouse Button	Toggle the I/O when pressed once. (i.e. OFF becomes ON and ON become OFF)

4. Our ladder program requires us to "push" the "Start" button momentarily. You can simulate this action by moving the mouse pointer to the "Start" label (or the LED lamp) and press the LEFT mouse button once and then release the button. The action starts!
5. At this time, notice that the relay "RUN" is latched ON and the timer "Duration" begins to count down from the value of 1000 every 0.1sec, and the Output #1-#8 are turning ON/OFF sequentially in a "running light" pattern. Sequencer "Seq1" in the "Ctr/Seq" column begins to count upward from 1 to 3 and then overflows to 0 and repeats continuously. For each step of the Sequencer, the corresponding Output will be turned ON. Our demo program will show a running light pattern starting from Outputs 1 & 8, then 2 & 7, 3 & 6 and 4 & 5 and then back to 1 & 8, 2 & 7.....
6. Now you should verify that the logic works as intended by observing the ladder diagram. You should notice that the "Run" labels in all circuits are highlighted as shown below:



7. The logic states of any I/O can be displayed on the ladder diagram directly. An Input, Output, Relay, Timer or Counter contact that is turned ON will have its label name highlighted in the ladder diagram. This feature helps greatly in debugging and understanding the logical relationship between each I/O. For example, from the above figure, we can see clearly that the "Self-latching" circuit for relay "Run" works as intended: when we first turn ON the "Start" input, "Run" will be energized and its contact which is parallel to "Start" will hold itself in the ON state, even if we subsequently turn OFF the "Start" input by releasing the button.
8. The timer coil "Duration", being connected in parallel to "Run" relay, will also be energized. However, its contact will only be closed after 100 seconds (when its present value counted to 0). To break the latched On "Run" relay, we must energize the "Stop" input momentarily to break the "power" flow. Try it now.
9. Let's restart the system by turning ON the "Start" input momentarily again. Next, we want to turn ON the "Manual" input. Move the mouse pointer to the "Manual" input and then press the right mouse button. "Manual" input will be "stuck at "ON" state even after you have released the right mouse button. Click on "Manual" button using the right mouse button again and it will be turned to OFF.
10. When "Manual" input has been turned ON, the running lights should stop. This is because the normally-closed contact of the "Manual" input in Circuit #2 is now turned OFF and the 0.5s clock pulse could not trigger the [AVseq] function anymore.
11. If you now left-click on the "Step" input, the running lights will move one step at a time in response to your mouse click. Observe the Seq1:x contact with respect to the counter value of Seq1 and the logic of this circuit become very clear instantly.
12. Observe that the timer "Duration" continues to count down every 0.1 second, and when it reaches 0, the "Duration" output coil label will be highlighted. You can use this timer to stop the program by connecting a N.C. "Duration" contact to Circuit #1. This is left as an exercise for you!
13. You can also observe the execution of the Custom Function #1 by clicking on the "View" button on the "Programmable Logic Simulator" screen and the following window will appear:



```
A= 0          B= 0          C= 0          D= 0          E= 0
F= 0          G= 0          H= 0          I= 0          J= 0
K= 0          L= 0          M= 0          N= 0          O= 0
P= 0          Q= 0          R= 0          S= 0          T= 0
U= 0          V= 0          W= 0          X= 4          Y= 0
Z= 0          HSC1= 0            HSC2= 0            HSC3= 0
```

CH#	1	2	3	4	5	6	7	8
ADC 1-8	0	0	0	0	0	0	0	0
9-16	0	0	0	0	0	0	0	0
DAC 1-8	0	0	0	0	0	0	0	0
9-16	0	0	0	0	0	0	0	0
PWM 1-8	0	0	0	0	0	0	0	0

Date: 2005/10/21
Time: 15:0:0
Day : Fri
LED ----------------

4 lines LCD Display

```
Cycle Count = 4
```

14. This screen shows the values of TBASIC variables A to Z and many other data. It also provides a simulated 4 lines LCD display that show case the action of the SETLCD statement contained in Custom Function #1. As you can see, the value of variable X used in the Custom function is also shown in the variable section. The value of X is converted by the STR$(X) statement into a string and displayed after the text string "Cycle Count = " at line #1, column #1 of the LCD area.

Summary

We have completed this hands-on session and have successfully created a simple ladder +BASIC program. We have also performed real time simulation to test the program's functionality. By now you would probably have a good appreciation of i-TRiLOGI's superb capability and ease of use and are ready to include i-TRiLOGI as an integral part of your programming needs.

جزء شرح إضافي Using i-TRiLOGI Sequencers

A sequencer is a highly convenient feature for programming machines or processes which operate in fixed sequences. These machines operate in fixed, clearly distinguishable step-by-step order, starting from an initial step and progressing to the final step and then restart from the initial step again. At any moment, there must be a "step counter" to keep track of the current step number. Every step of the sequence must be accessible and can be used to trigger some action, such as turning on a motor or solenoid valve, etc. As an example, a simple Pick-and-Place machine that can pick up a component from point 'A' to point 'B' may operate as follow:

Step #	Action	**Step #**	Action	**Step #**	Action
0	Wait for "Start" signal	3	Retract arm at point A	6	Open gripper
1	Forward arm at point A	4	Move arm to point B	7	Retract arm at point B
2	Close gripper	5	Forward arm at point B	8	Move arm to point A

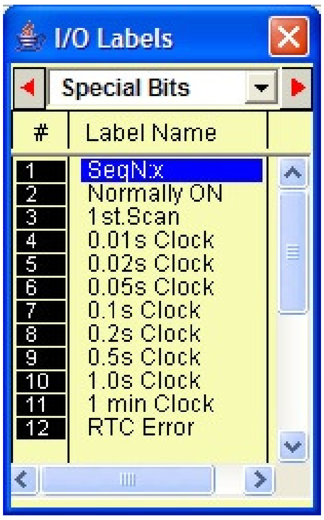
i-TRiLOGI Version 5 supports eight sequencers of 32 steps each. Each sequencer uses one of the first eight counters (Counter #1 to Counter #8) as its step counter. Any one or all of the first eight counters can be used as sequencers "Seq1" to "Seq8".

To use a sequencer, first define the sequencer name in the Counter table by pressing the <F2> key and scroll to the Counter Table. Any counter to be used as sequencer can only assume label names "Seq1" to "Seq8" corresponding to the counter numbers. For e.g. if Sequencer #5 is to be used, Counter #5 must be defined as "Seq5". Next, enter the last step number for the program sequence in the "Value" column of the table.

Construct a circuit that uses the special function "Advance Sequencer" [AVSeq]. The first time the execution condition for the [AVseq] function goes from OFF to ON, the designated sequencer will go from inactive to step 1. Subsequent change of the sequencer's execution condition from OFF to ON will advance (increment) the sequencer by one step. This operation is actually identical to the [UPctr] instruction.

The upper limit of the step counter is determined by the "Set Value" (SV) defined in the Counter table. When the SV is reached, the next advancement of sequencer will cause it to overflow to step 0. At this time, the sequencer's contact will turn ON until the next increment of the sequencer. This contact can be used to indicate that a program has completed one cycle and is ready for a new cycle.

Accessing individual steps of the sequencer is extremely simple when programming with i-TRiLOGI. Simply create a "contact" (NC or NO) in ladder edit mode. When the I/O window pops up for you to pick a label, scroll to the "Special Bits" table as follow:



The "Special Bits" table is located after the "Counters" table and before the "Inputs" table. Then click on the "SeqN:x" item to insert a sequencer bit. You will be prompted to select a sequencer from a pop-up menu. Choose the desired sequencer (1 to 8) and another dialog box will open up for you to enter the specific step number for this sequencer. Each step of the sequencer can be programmed as a contact on the ladder diagram as "SeqN:X" where N = Sequencers # 1 to 8. X = Steps # 0 - 31.

e.g. Seq2:4 = Step #4 of Sequencer 2.

Seq5:25 = Step #25 of Sequencer 5.

Although a sequencer may go beyond Step 31 if you define a larger SV for it, only the first 32 steps can be used as contacts to the ladder logic. Hence it is necessary to limit the maximum step number to not more than 31.

Special Sequencer Functions

Quite a few of the ladder logic special functions are related to the use of the sequencer. These are described below:

Advance Sequencer - [AVseq]

Increment the sequencer's step counter by one until it overflows. This function is the identical to (and hence interchangeable with) the [UpCtr] function.

Resetting Sequencer - [RSseq]

The sequencer can also be reset to become inactive by the [RSseq] function at any time. Note that a sequencer that is inactive is not the same as sequencer at Step 0, as the former does not activate the SeqN:0 contact. To set the sequencer to step 0, use the [StepN] function described next.

Setting Sequencer to Step N - [StepN]

In certain applications it may be more convenient to be able to set the sequencer to a known step asynchronously. This function will set the selected sequencer to step #N, regardless of its current step number or logic state. The ability to jump steps is a very powerful feature of the sequencers.

Reversing a Sequencer

Although not available as a unique special function, a Sequencer may be stepped backward (by decrementing its step-counter) using the [DNctr] command on the counter that has been defined as a sequencer. This is useful for creating a reversible sequencer or for replacing a reversible "drum" controller.

Other Applications

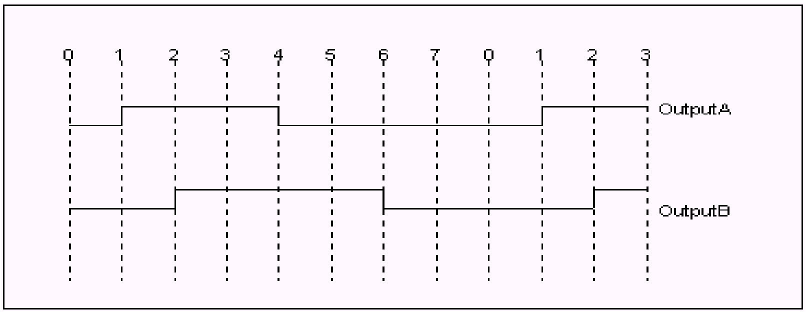
a. Driving Stepper Motor

A sequencer may be used to drive a stepper motor directly. A two-phase stepper motor can be driven by four transistor outputs of the controller directly (for small motors with phase current < 0.5A) or via solid-state relays. The stepper motor can be driven using a sequencer that cycles through Step#0 to Step#3 (full-step mode) or Step#0 through Step#7 (half-step mode). Each step of the sequencer is used to energize different phases of the stepper motor. A clock source is needed to drive the stepper motor through its stepping sequence. The stepping rate is determined by the frequency (which is equal to 1/period) of the clock source.

Clock pulses with periods in multiples of 0.01 second can be generated easily using the "Clk:.01s" bit and an [Upctr] function. For e.g., to generate a clock source of period = 0.05s, use "Clk.01s" to feed to an [Upctr] counter with Set Value = 4. The counter's contact (completion flag) will be turned ON once every 5 counts (0,1,2,3,4), which is equivalent to a 0.05 sec. clock source.

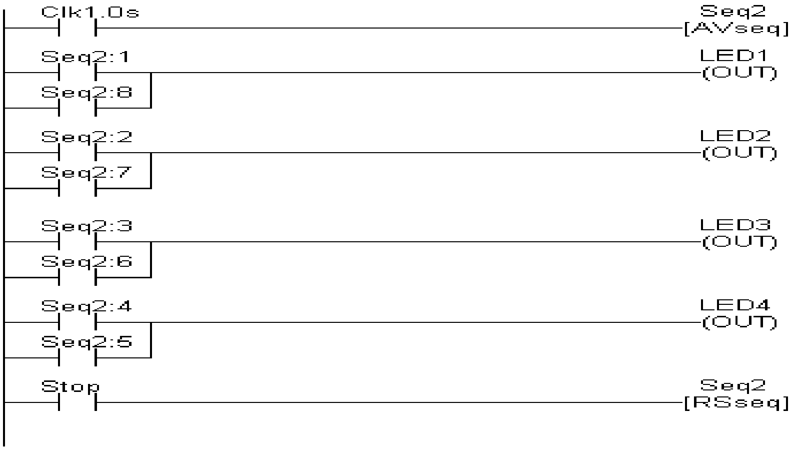
b. Replacing a Drum Controller

A drum controller can be replaced easily by a sequencer if the timing of the drum's outputs can be divided into discrete steps. Assuming a drum controls two outputs with the timing diagram shown in the following figure: This can be replaced by an 8-step sequencer. Step 1 (e.g "Seq1:1") turns ON and latch Output A using [Latch] function, Step 2 turns ON and latch Output B, Step 4 turns OFF Output A using the [Clear] function, and Step 6 turns OFF Output B. All other steps (3,5,7,0) have no connection.



Program Example

Assume that we wish to create a running light pattern which turns on the LED of Outputs 1 to 4 one at a time every second in the following order: LED1, LED2, LED3, LED4, LED4, LED3, LED2, LED1, all LED OFF and then restart the cycle again. This can be easily accomplished with the program shown in Figure 6.9.



The 1.0s clock pulse bit will advance (increment) Sequencer #2 by one step every second. Sequencer 2 should be defined with Set Value = 8. Each step of the sequencer is used as a normally open contact to turn on the desired LED for the step. A "Stop" input resets the sequencer asynchronously. When the sequencer counts to eight, it will become Step 0. Since none of the LED is turned ON by Step 0, all LEDs will be OFF.