Ti-PLC

User Manual

Version 1.0 β, by Kuashio
kuashio@hotmail.com

Abstract

This document is a user manual and reference guide for Ti-PLC, a PLC simulator and emulator program for the Ti-89 graphing calculator from Texas Instruments.

I should say that the program is rather easy to use, and there's practically no need for a manual if you are experienced in PLC programming. However, there are many details the user should know in order to use it properly.

Contents

Introduction

Ti-PLC is a PLC Simulator/Emulator program for the Ti-89 made for learning purposes: for the author (learn TIGCC) and the user (learn about PLCs). This program has a variety of controls and tools that are explained in detail below.

If you are already familiar with PLCs, you may skip the first part of this document, which explains the fundamentals of PLCs.

PLC Basics

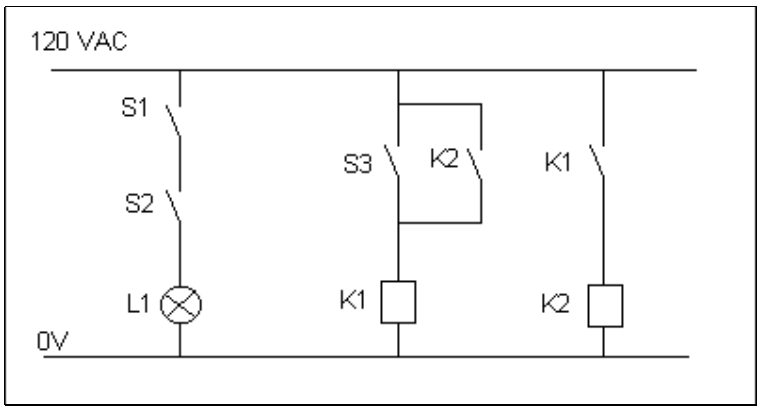
Briefly, a PLC (Programmable Logic Controller) is a digital controller designed for industrial automation. It meets all the requirements for industrial work.

Most PLCs have the following components:

- A microprocessor unit that executes any program stored in its non-volatile memory.
- A WDT (WatchDog Timer), which is a counter that starts counting when a program iteration begins and resets when it ends. The carry bit of this counter triggers a system protection routine which may consist on resetting program execution. The purpose of this counter is to react in case the microprocessor unit doesn't respond. Safety is a very important issue in the industry.
- A digital input port, where logical states may be represented by the presence or lack of voltage in their terminals.
- A digital output port, generally with relay-controlled power outputs where logical states may be represented by an energized or not-energized relay coil, which supplies power to the output terminals. In some cases, the relay's contacts have one common terminal connected to the logical "1" voltage line. This makes outputs always work under the same logic as inputs.
- A set of general purpose counters, which are addressable from the program executed by the PLC. These counters are simply registers with the size of the microprocessor's data bus. They store integers.
- A set of general purpose timers, also addressable from user code. As for counters, these registers have the size of the system's data bus, and store integers, which represent the quantity of specific units of time that have passed since the count has begun. These time units are often specified by the manufacturer as a parameter of the PLC and its timers.
- A register of general purpose flags. These flags store program states defined by the programmer (not the manufacturer).

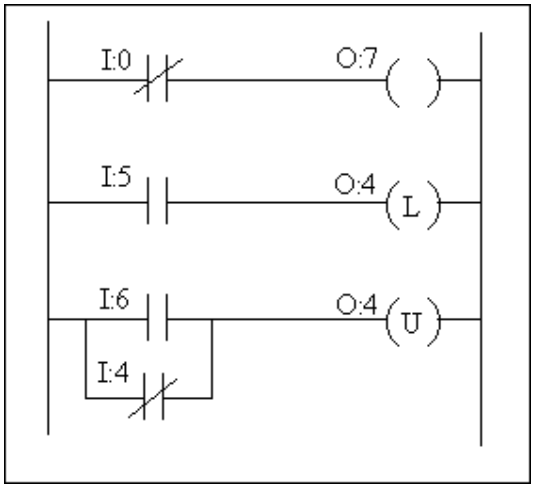
Ladder Programming

Ladder diagrams are very similar to control diagrams made with electrical circuits, in which a horizontal upper line is drawn to represent a voltage line, then a circuit (made from switches and loads), and finally a lower horizontal line of 0V.
When the circuit is closed, the load associated to the circuit gets energized as shown below.



A program represented as a ladder diagram works in a similar way. In this case, a left vertical line is drawn to represent voltage, then a boolean expression (made of states of hardware elements which can be true or false), and a right vertical line that represents the 0V line.

A ladder diagram is made of rungs (equivalent to lines of code), and each rung is divided in two parts: a Condition and an Action (equivalent to an if(...) then{...} construct).
The condition is equivalent to the switch circuit, and the action is equivalent to the loads as shown below. An action happens only if its condition is true.



Note: The circuit and ladder diagram above are not equivalent.

Execution Sequence

Since it's an industrial control device, a PLC is an event driven controller, which means that it must be ready for any situation to happen in its inputs (and internal registers) and modify its outputs (and internal registers) when it is needed.

This makes it necessary for the PLC to work on infinite cycles. The standard for these cycles is the following execution sequence:

- Freeze Inputs
- Run Ladder
- Update Outputs

In most cases, a Watchdog timer is supervising the PLC's operation.
Each of these iterations is called a Scan.

Well, that's it for my PLC introduction. If you want to read more on PLCs, including tutorials, go to http://www.thelearningpit.com/

Ti-PLC

This project is divided in two main modules: A Simulator made in TIGCC, and an Emulator made also in TIGCC, communicated with the real world through a simple TTL interface circuit made with a BASIC Stamp II module, a PIC-based Single Board Computer with 16 I/O pins.

For more on BASIC stamps, go to http://www.parallaxinc.com/

Getting Started

I am writing this document prior to making the .89g file, but surely (and hopefully) all you have to do to install it is send the group file to your Ti-89.

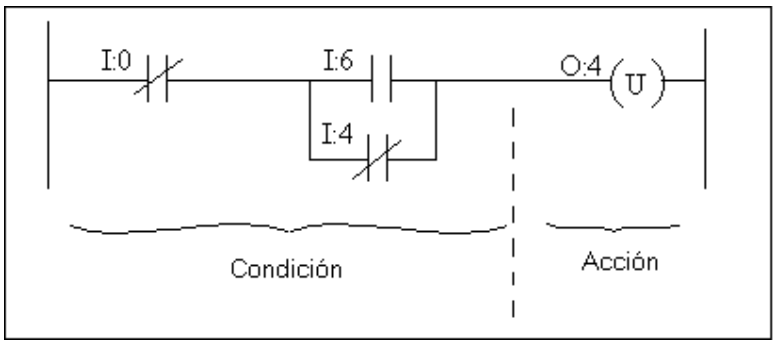
When you enter tiplc() in the command line, your Ti-89 HW2 should run my program with no problem.

The Generic PLC

Here are the addresses for the internal registers of the generic PLC simulated and emulated by Ti-PLC. This is the address standard used here.

Registers	Addresses
8 Inputs	I:0, I:1, I:2, I:3, I:4, I:5, I:6, I:7
8 Outputs	O:0, O:1, O:2, O:3, O:4, O:5, O:6, O:7
8 Flags	B:0, B:1, B:2, B:3, B:4, B:5, B:6, B:7
8 Counters	C:0, C:1, C:2, C:3, C:4, C:5, C:6, C:7
8 Timers	T:0, T:1, T:2, T:3, T:4, T:5, T:6, T:7

For the execution of a ladder diagram, it is assumed that a rung is divided in two parts. Again: the condition and the action. In Ti-PLC, it is assumed that the action is represented by the symbol located at the LAST cell. Anything located before the rightmost cell is considered as part of the condition.
A condition may have branches as shown below.

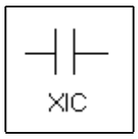
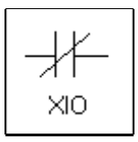
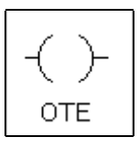
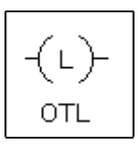
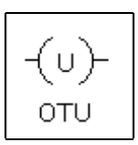
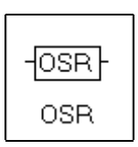
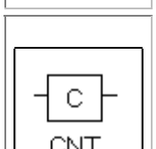
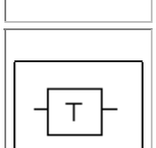
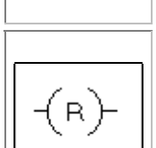
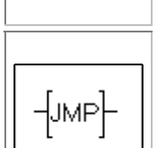
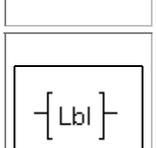


Please excuse the labels in spanish. I'm sure you understand them, though.

In the rung shown above, O:4 would respond to the following C statement:

```
If ( !(I:0) & ((I:6) | !(I:4)) ) then { O:4 = 0}
```

Ti-PLC Symbols

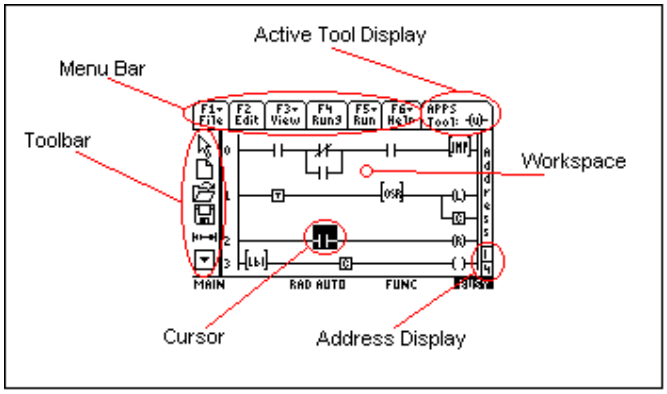
Mnemonic	Meaning	Part of Rung	Addressable	Description
	Examine If Closed	Condition	I, O, B	Examines whether the addressed bit is set, in which case, it closes its terminals (true). Otherwise, it opens them (false).
	Examine If Open	Condition	I, O, B	Examines whether the addressed bit is reset, in which case, it closes its terminals (true). Otherwise it opens them (false).
	Output Energize	Action	O, B	Sets the addressed bit whenever the rung condition is true. Otherwise, it resets the addressed bit.
	Output Latch	Action	O,B	Sets the addressed bit when the rung condition is true.
	Output Unlatch	Action	O,B	Resets the addressed bit when the rung condition is true.
	One-Shot Rising	Condition	B	Closes its terminals (true) for one single scan when it detects a rising edge in its left terminal. Otherwise it opens its terminals (false). Its state is stored in the addressed bit.
	Counter	Both	C	Condition: Closes its terminals (true) when the addressed counter equals the value entered in its properties. Action: Increments or decrements the addressed counter in 1 (depending on the direction set in its properties) when it detects a rising edge in the rung condition.
	Timer	Both	T	Condition: Closes its terminals (true) when the addressed timer equals the value entered in its properties. Action: Increments the addressed timer in 1 when the rung condition is true. Otherwise, it resets the timer if it is not set as a retentive timer in its properties.
	Reset	Action	C, T	Resets the addressed counter or timer to 0 when the rung condition is true. To reset a bit, use OTL.
	Jump	Action	None	If the condition is true, the execution jumps to the rung where the label refered to is located.
	Label	Condition	None	A label's terminals are always closed. It doesn't actually do anything. It's only purpose is to identify the rung when a JMP symbol is executed.

Ti-PLC Description

Ti-PLC has two screens: The Ladder Editor and The Animation.

The Ladder Editor Screen

This is the screen you'll see when you start the program (after the splash screen, which is actually the Animation screen). It has the following components:



Cursor
This should be clear already. Highlights a location in the workspace.

Active Tool Display
This is where the active tool is displayed. Uhh, yeah (?).

Address Display
This is where the address for the highlighted symbol (if any) is displayed.

Menu Bar

- File
  - Quit
- Edit (Not yet implemented)
- View
  - Toggle Rung Numbers (In case you want to hide them)
  - Toggle Grid (Toggles the grid, duh)
- Rung (Not yet implemented)
- Run
  - Simulate
  - Emulate
  - Settings... Brings up a dialog box that prompts you for the baud rate (for the emulator), and the watched counters and timers (for the animation).
- Help
  - Help A brief user guide with the basics.
  - About

Toolbar

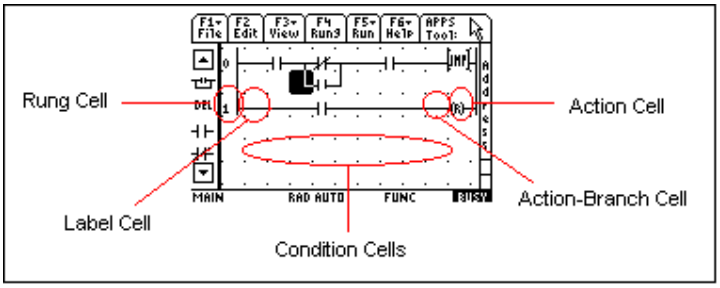
This toolbar has all the tools and symbols you'll need to make a ladder diagram. Here's a brief description of what they do.

Tool	Description
Arrow	It brings up the properties dialog box for the symbol it was clicked over. You are free to enter the properties you want in this dialog box. However, you'll have to respect the address standard (I,O,B,C,T):(0,1,2,3,4,5,6,7).
New	It only clears the workspace. Since you cannot save yet, you'll lose all your work if you use this tool.
Open	Not yet implemented.
Save	Not yet implemented.
New Rung	Places a new rung above the clicked rung.
New Branch	Places a new inclusive branch between two clicked cells. It works from left to right. If you click twice on the action branch cell (the action cell's left neighbor), you'll get an action branch in case you want to run more than one action in a rung.
Delete	Deletes the symbol it's clicked over. If you click over a rung number, it deletes the whole rung. If you want to delete a branch, you'll have to delete the whole rung (zurrie!).

The rest of the tools are used to insert their related symbols into the ladder under construction. If you try to insert a symbol over an existing symbol, Ti-PLC will warn you that the original symbol will be replaced. When you insert a new symbol, its properties dialog box will pop so you can enter its properties.

Workspace

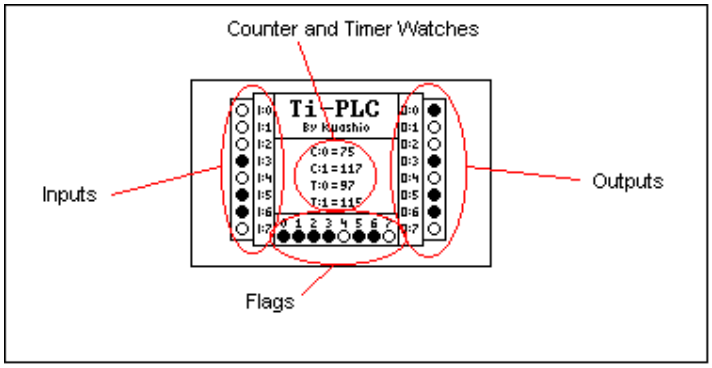
Here's where you work. The ladder, the grid and rung numbers are displayed here. The workspace is made of cells. There are 11 cells in a rung, and there are some cells that are mentioned by their names in this manual. So here are their names:



Cell Number	Name	Purpose	Remarks
1	Rung Cell	Display the rung number.	Nothing can be inserted here.
2	Label Cell	Place labels here.	Condition symbols may be inserted here. Labels can only be inserted here.
2 through 10	Condition Cells	Place the condition here.	Condition symbols may be inserted here.
10	Action Branch Cell	Start an action branch here.	Condition symbols may be inserted here.
11	Action Cell	Place the action here.	Action symbols may be placed here.

The Animation Screen

This is the screen you'll see as a splash screen and during your simulation and emulation. It serves as a monitor of the internal PLC's states. It has the following components:



Bit Watches

These are the binary watches represented by circles, where a set bit is represented by a black circle, and a reset bit is represented by a white circle.

Register Watches

These are the integer watches shown as decimal numbers. Note that there are 8 counters and 8 timers available, but there are only 2 counters and 2 timers watchable. You can set which counters and timers are going to be watched in the Run→Settings dialog box.

Simulation and Emulation

The animation displays the PLC's states for both, the simulator and the emulator. Here are the main differences between them:

In the...	Inputs are taken from...	Outputs are delivered to...
Simulator	Keyboard keys "0" through "7" in a bistable behavior.	Nowhere, actually.
Emulator	The interface circuit's physical input pins.	The interface circuit's physical output pins.

Simple Steps to use Ti-PLC

1. Enter tiplc() in your Ti-89 command line.

2. Make your ladder diagram. Please read the assumptions and notes below.

3. Run your program from the "Run" menu. You may run it as a simulation or as an emulation. For emulation, you need an external circuit, though.

4. The input for the simulator is taken from the keyboard numbers 0 through 7, and the output is shown in the animation.

   The input and output for the emulator are the ones present in the external interface circuit.

   The PLC registers (I,O,B,C(2),T(2)) are always watched in the animation.

5. You can't save for many reasons (see the note about future versions below). But you don't have to save if you want to leave Ti-PLC, because it uses an allocated data structure for the buffer, so you will see the same ladder when you run Ti-PLC again.

6. To stop the animation and return to the ladder editor, just press Esc.

Ti-PLC Assumptions

Have you seen "Under Siege 2"? In that movie, some bad guy defines "Assumption" (you may know what I mean), so here are all the assumptions made in Ti-PLC for a program to run properly.

- Actions must be placed at the rightmost cell in a rung. This is what I call the "Action Cell". Anything not placed in the action cell is considered as part of the condition (located in the “Condition Cells”). If you need more than one action to be made in a rung, you may add them using branches. To insert an action branch, just click twice over the action branch cell.

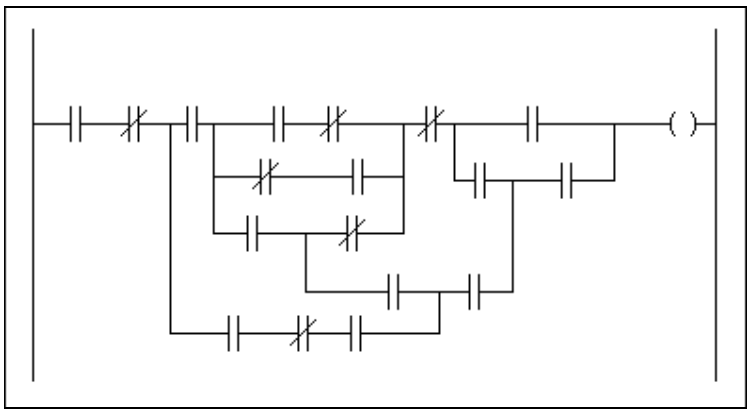
- The LBL symbol must be placed at the beginning of a rung. Specifically, it must be placed at the first label cell. Otherwise it won't work. Yes, I know: It's just the same as if rung numbers were used. But a label is more versatile in case you insert a new rung before the labeled rung: You won't have to change the reference in the JMP symbol.

- There is (still) no watchdog here! So be careful with the JMP symbol. If you stay within the ladder without finishing it, the inputs and outputs won't be updated, so your program won't communicate with the real world! This means there’s a hazard related to the use of the JMP symbol and the lack of a watchdog. I included the JMP symbol just to make it possible to skip some rungs in a given case, therefore I would only direct a JMP symbol to a rung below and never above the current rung (However, it is possible. But you have know what you're doing).

- There's no need for an END rung in Ti-PLC.

Notes from the Author

- Don't try to make funny branches, there are still some bugs with complex branches. If you think it over, you may not need a very complex expression in one rung. By "complex", I mean something similar to the following rung:



- Also, if you want to delete a branch, you'll have to delete the whole rung (zurrie bout that!).

- I tested the emulator in a Ti-89 HW2. I am only certain that it works there.

- Always watch your timers and counters in the animation. That's why I made it possible.

- Build and use the interface circuit! It's worth the effort.

- Don't get mad at me because you can't save. You actually Can Save if you think about it: You can always save VTI states!

TTL Interface

WARNING: This part is intended for electronics technicians, engineers, electronics literate people, phreakers, and people who don't mind screwing up their calcs because they don't have experience on reverse engineering (almost none involved). I am NOT responsible for your mistakes.

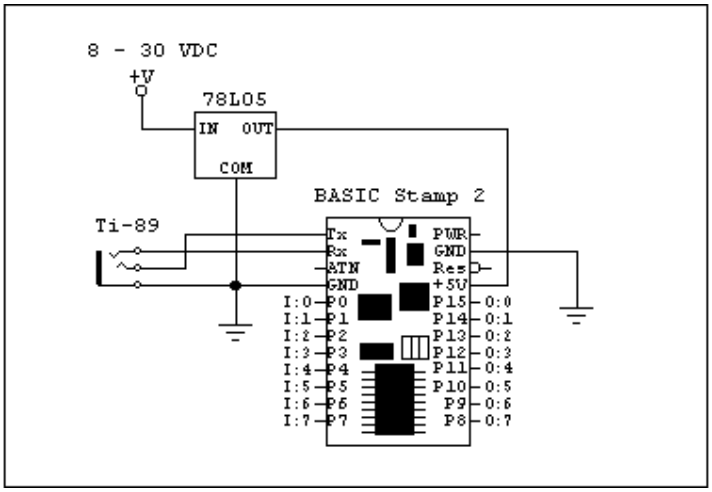
Well! Having said that, and for all those who are still reading...

I made my interface using a BASIC Stamp II module from http://www.parallaxinc.com However, you may build it with any other chip (HC11, 386, PIC, dynamic TTL, etc.). The only requirement is that it works with my "data bounce" protocol. This protocol is rather easy to understand. Here's the general algorithm:

1. Wait for an RS-232 serial byte (1 start bit, no parity).
2. Store the received byte in the output register (physical outputs).
3. Store the input register (physical inputs) in the serial output buffer.
4. Flush the serial output buffer through the RS-232 line.

The Ti-89 and Ti-PLC take care of the rest. Obviously, your circuit must use the same baud rate as the one specified in the "Run→Settings..." dialog.

If you want to make exactly the same interface I made, here's the schematic.



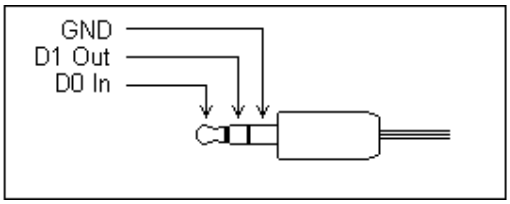
And here's the code for the BS2. For a good understanding of the following code, go check the manual for the BS2 module at http://www.parallaxinc.com .

```
serdata Var byte

     dirs=$FF00

     Start:
          serin 16,16468,65535,start,[serdata]
          OUTH=serdata REV 8
          serout 16,16468,[INL]
     goto start
```

The only requirement still missing is the linkport pinout, which is shown below.



The terms “In” and “Out” are from the Ti-89’s point of view. So those two bits called D0 and D1 are part of the Ti-89/HW2‘s register shown below. You can see the register’s address.

0x60000C DBus configuration / status (IE = Interrupt Enable)															
Control								Status							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AE	LD	LTO		CLE	CAIE	CTX	CRX	SLE	STX	SRX	SLI	SA			

DBus Configuration Register

0x60000E Link Data — Send / receive data through the link port.															
Low								High							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				D1 In	D0 In	D1 Out	D0 Out	RX / TX buffer							

Link Register

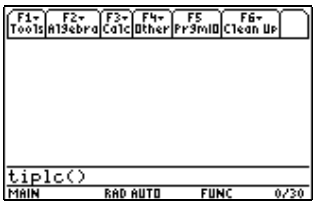
Tutorial

Making a "Hello World!" in Ti-PLC

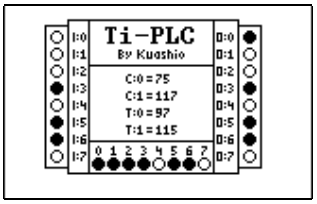
Alright, let's make a ladder program in Ti-PLC! Our helloworld will work as follows:

- Counter C:4 will count how many times I:2 and I:3 are set; and O:3 will respond to this AND condition.
- Timer T:1 will count (retentively) how much time I:4 or I:5 are set; and O:5 will respond to this OR condition.
- Flag B:3 will respond to I:6 and the inverted state of I:7.
- Flag B:3 will reset timer T:1 and counter C:4.
- Flag B:3 will be set when counter C:4 equals 5.

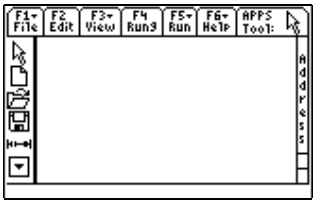
Here's the procedure to follow step by step:



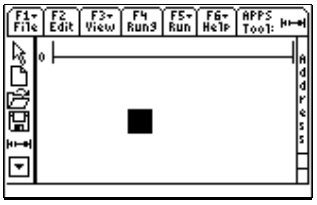
Enter "tiplc()" in the command line from the main folder.



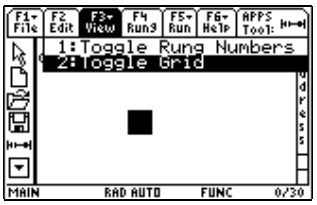
The splash screen will show.



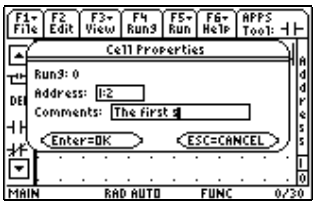
This is the empty editor screen.



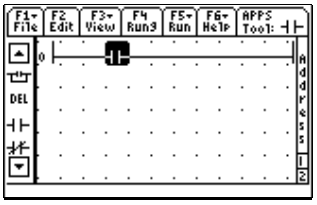
Using the arrow keys, move the cursor in the toolbar and select the "New Rung" tool pressing Enter. Place the cursor anywhere in the workspace and place the new rung pressing Enter. A new rung (numbered "0"), will appear.

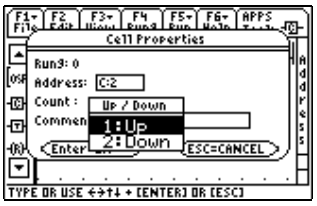


In case you want to use a grid, you can toggle it using the View menu. You can also toggle the rung numbers from the View menu.

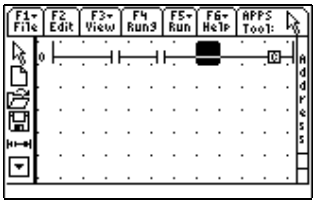


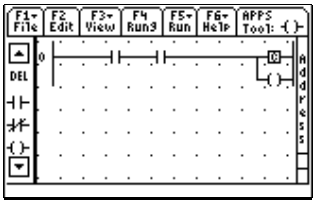
Select the XIC symbol tool from the toolbar and place the symbol somewhere in the condition cells. The Cell Properties dialog box will pop. Since this is a new symbol being entered, the default properties are set. Change the address to I:2 and add some comments if you want to.

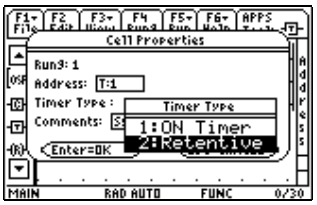

The symbol will be placed where you requested. Notice the address shown at the address display. Insert another XIC symbol in the same rung (say two cells to the right) without selecting the XIC symbol tool again. Set I:3 as the address for this new symbol.



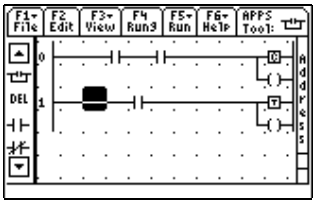
Select the CNT symbol tool from the toolbar and place the symbol in the action cell. This time, the Cell Properties dialog box asks you for the direction of the count (select an Up counter). Change the address to C:2 and add some comments if you want.

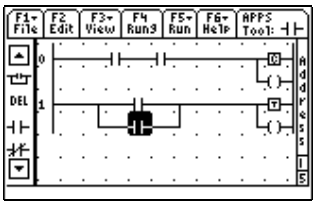

Your ladder should look like this by now. If you want to modify any properties to the symbols, you can do it using the Arrow tool. Go change the counter's address to C:4.

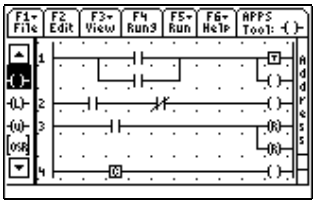

Select the New Branch symbol tool from the toolbar and hit Enter twice in the Action Branch cell. An action branch will appear. Place an OTE symbol as a parallel action addressed at O:3. When the cursor is visible, you can hide it pressing Esc once (Don't press Esc twice, or Ti-PLC will ask if you want to quit). Your diagram should look like the picture.



Insert a new rung using the New Rung tool and pressing Enter below the existing rung. If you make a mistake, you can delete entire rungs using the Del tool and pressing enter over the Rung cell; to delete symbols, do it over their locations. In the new rung, enter a TMR symbol as an action (at the action cell). Address it to T:1 and select a retentive timer. Add comments if you want to.

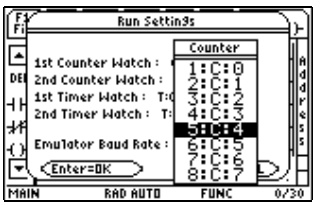

Insert an action branch and then an OTE symbol over the new action branch, addressed to O:5. Insert an XIC symbol in the condition cells, addressed to I:4. Now, select the New Branch tool again and place the cursor over an empty cell located to the left of the new XIC symbol (where you want your branch to begin) and press Enter. Your ladder should look like the picture suggests.


Now move to another empty cell located to the right of the new XIC symbol (where you want your branch to end) and press Enter. A new branch will appear. Now place a new XIC symbol somewhere in the new branch and address it to I:5. Your ladder should look like the picture.

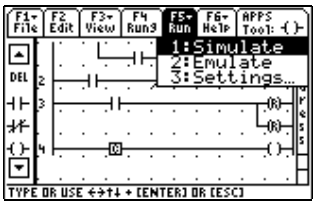

Add the rungs and symbols shown in this picture. When you insert rungs exceeding the workspace's display capacity, you can always scroll up and down using the arrow keys on the workspace.

Here are the expected addresses and values for the whole ladder diagram:

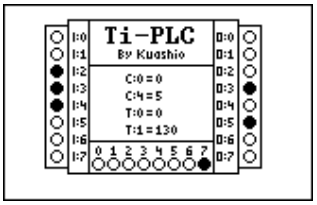
Rung	Symbol	Address	Value
0	XIC	I:2	
0	XIC	I:3	
0	CNT	C:4	Up
0	OTE	O:3	
1	XIC	I:4	
1	XIC	I:5	
1	TMR	T:1	Retentive
1	OTE	O:5	
2	XIC	I:6	
2	XIO	I:7	
2	OTE	B:3	
3	XIC	B:3	
3	RES	C:4	
3	RES	T:1	
4	CNT	C:4	5
4	OTE	B:7	



Select the watched counters and timers from the Run→Settings dialog box.



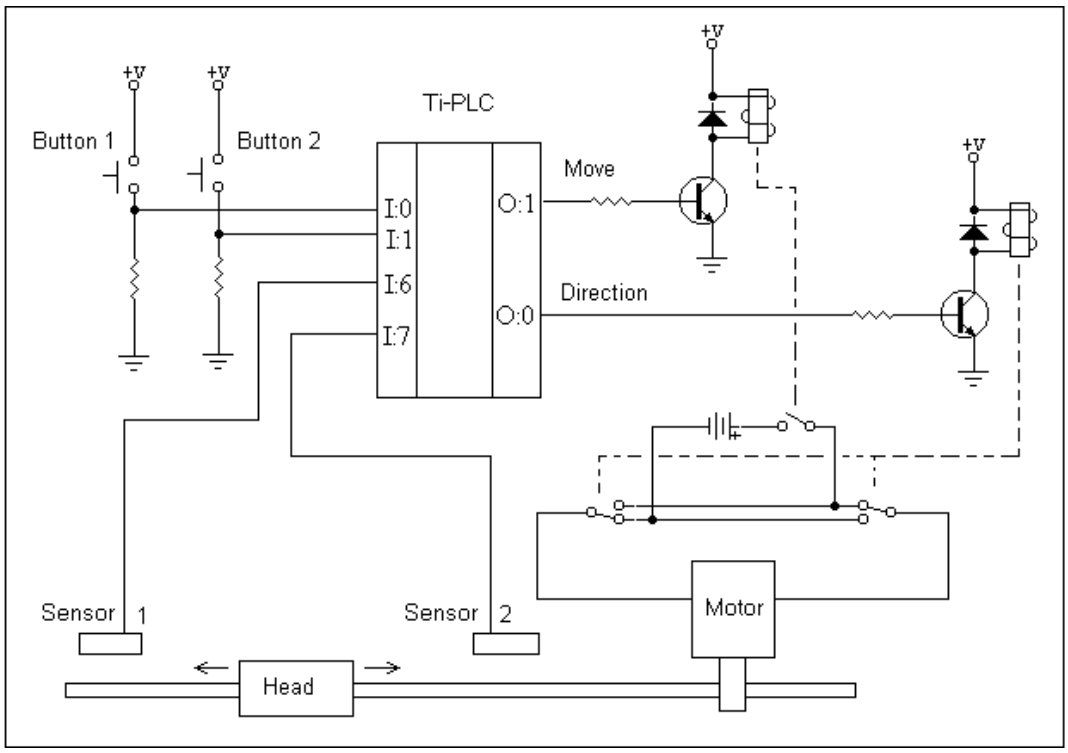
Save your VTI state image (if you're using your PC) and start the simulation from the Run menu.



This is the animation screen. Toggle the input values using the number keys. Play with the inputs and see what happens. Does the PLC behave as you expected? If not, debug your program (or mine!).

Test Yourself: Example System for Emulation

OK, now that you are familiar with the ladder editor, I challenge you to make the following system actually work. Write down your programs in paper first.



Do you understand the diagram? I'm assuming that you do. This is a very simple circuit and the labels make it self-explainatory.

Note that Ti-PLC is shown as a single TTL PLC here because that's what it is as far as the rest of the circuit is concerned. This means that Ti-PLC includes a Ti-89 (with tiplc() running) and the interface circuit.

The lower part is supposed to be an old printer's mechanical system, which must have the following:

- ✓ A DC motor that controls the position of the printer head.
- ✓ A binary sensor on each end of the head's path.

For the following tasks, let Button1 = "Start" and Button2 = "Stop". The programs must run only when the system is started. If the system is stopped, the head should hold still. If both buttons are pressed, priority should be given to the "Stop" button.

- Make a program that makes the head go to sensor1, then go to sensor2 and stop there.
- Make a program that makes the head go from sensor1 to sensor2 and back again, say... 5 times (I assume that you know that I'm expecting you to use a counter).
- Make a program that makes the head go to sensor1 and stay there for a little while, then go to sensor2 and stay there for double the time it stayed on sensor1, then do it again and again and again... forever. Use timers.

For the following tasks, let Button1 = ”Go to Sensor 1”, and Button2 = ”Go to Sensor 2”. If both buttons are pressed, priority should be given to the “Go to Sensor 1” button.

- Make a program that makes the head move towards the desired sensor while the button is pressed. When no button is pressed, the head holds still. When the head reaches its goal, it holds still.
- Make a program that makes the head move towards the desired sensor from the moment the button is pressed. When the button is released, or if no button is pressed, the head must finish The Last Command. When the head reaches its goal, it holds still.

Frequently Asked Questions

Q: Where's the END symbol?
A: You don't need it in Ti-PLC.

Q: How can I insert a branch?
A: Just select the Branch tool and click over an empty cell in a rung (where you want the branch to begin). Ti-PLC is waiting for you to click over the cell you want the branch to end. When you click on a cell located at least two cells to the right of the branch beginning (and before the action cell), the branch is inserted. If you click somewhere else, the branch beginning is stored there. If you never set a branch end, nothing happens.

Q: How can I delete a branch?
A: You're assuming that you can. But you can't. Not yet. You'll have to delete the whole rung. Sorry about that! E-mail me asking me to add that.

Q: How can I delete a rung?
A: Just select the DEL tool and click over the rung number cell (where the rung begins).

Q: How can I insert multiple actions in a rung?
A: You'll have to use action branches, which are inserted by selecting the Branch tool and clicking twice over the action branch cell.

Q: My program won't run some actions. What am I doing wrong?
A: Many things could be wrong.

- Those actions may be outside the action cell (this may happen with counters and timers).
- Some addresses may be incorrect.
- You may not be using the address standard.

Q: What's with the address standard?
A: The standard for an address is a 3-character string, where the first character indicates the register and the third one indicates the address. The middle character is irrelevant (it's simply ignored), but ":" is suggested. For example:

To refer to...	The address would be...
Input # 6	`"I:6"` or `"I/6"`
The last output	`"O:7"` or `"O.7"`
The first flag	`"B:0"` or `"B 0"`
The third counter	`"C:2"` or `"C-2"`
Timer # 5	`"T:5"` or `"T*5"`

Q: I can't save my ladder! Is there something I can do?
A: Of course you can't save! this is only a β version. And yes, there's something you can do: You can save a VTI state with your ladder diagram loaded. Of course, this would strictly require Ti-PLC to run on your PC with VTI (unless somebody finds a way to load a VTI state image into the Ti-89, which would RULE... anyone?).

Q: What are those 8 LEDs (numbered from 0 to 7) placed horizontally at the bottom of the PLC in the animation?
A: They are the flags B:0 through B:7.

Q: How can I convert the units stored in timers to seconds?
A: Since timers are actually counters without the OSR feature (actually, counters are timers with an OSR at their inputs, and they store signed integers), you'll have to convert those units using the "trial and error" technique considering this: Having a large ladder, your timers will take longer than having a small one.

Q: Why does the Ti-89 seem to be at the main folder in your screenshots?
A: Whoa! Your attention to details is above normal. All of the screenshots taken for this manual came from VTI, and it is less complicated to work with an empty image of the calculator in the main folder during development. Let's just pretend the calc is in the "tiplc" folder, ok?

Q: Do you have a life, or did you make this program to call people's attention?
A: Yes, I have a life. Some attention would be nice, though.

Q: How can I ever thank you for such a lovely program?
A: Just send me your comments to kuashio@hotmail.com

Q: Are there any future versions coming?
A: Only if you write to me.

A word on Future Versions

This is the last you'll ever see of this program unless I receive enough requests to upgrade it.

The only thing that I still consider missing in this program is file management (Open & Save), but I still lack the needed VAT knowledge. If anyone has suggestions, send them to me, although I think I have enough documentation in the TIGCC help file, so I just need a couple of free afternoons to learn all about it. But then again, I won't work on it unless I get some proof that my work is useful (email feedback).

This is my first TIGCC program and I should say that I noticed something that worried me: I've been used to code in C for PCs on DOS and Linux. Since these systems (PCs) work with a huge amount of available RAM, and their RAM resets when you turn them off, the memory-leakage issue is not often all that important. However, since the Ti-89 is always on, its limited RAM doesn't reset and memory-leakage becomes important. I'm an Electronics Engineer AND a programmer, so all this made me a better programmer yet. So I guess this was good for me.

Still to do...

- A Clipboard. This would ease the process of building a ladder diagram.
- A File Manager. This would make it possible to save and load ladder files.
- Upgrade the Delete tool. This would allow you to delete a branch properly.
- A Watchdog Timer. This would clear the JMP hazard mentioned above.
- Compatibility with the Ti-92+. This will take me a week or so.
- Anything else you ask me to add in your email.

Legal Stuff

About Ti-PLC

Ti-PLC (do I NEED to say Beta?) v1.0, first public release (yay!)
Made by Kuashio (kuashio@hotmail.com)
http://www.angelfire.com/geek/kuashio

License

- This is a kuashioware program, which means that you only have to send me comments, bug report, or suggestions by email.
- You may distribute it as long as you don't sell it.
- You may use my code (if you understand it! hehehe). I appologize for the (almost) lack of comments. Don't forget to include me in the credits.
- This project is intended for students.

Disclaimer

Since you didn't actually pay ANYTHING for this program, the author is NOT responsible in ANY way for ANYTHING that happens to ANYTHING as a result of using ANYTHING included in tiplc.zip, or tiplc.zip itself.

Credits

Kuashio,	Concept, code, electronics (forward and reverse) engineering, breadboarding, soldering, coffee drinking, testing, preachy talk to students, documentation, sleepless nites, etc..
Josh Christensen,	Provider of his fabulous serial RS-232 driver (sdriver.zip). Thanks for this wonderful idea, Josh!
Zeljko Juric,	Provider of the show_picvar() function to display bitmaps. Thank you for such a lovely documentation of TIGCC.
Rusty Wagner,	Provider of VTI. Thank you, Rusty! Without Vitual TI, the development of this project would've been *virtually* impossible.
André Felix Miertschink,	Provider of VTI Capture, a nice program for animated and static image capture. Thank you very much.
Kevin Kofler,	Provider of autoaoff(), a little ASM program that turns off the alpha-lock for dialog boxes. I know there are many people like me who aren't exactly thankful to lovely lovely TI for this "smart feature". So this program is a great escape and you only have to run it once. Thank you, Kevin , I'll be forever grateful to you for this excellent patch.
The TIGCC staff,	Providers of TIGCC-SDK. Thanx guys.
My students,	Beta testers (The ones who actually used it). I hope this helped.

About the Author

The author is originally from Tegucigalpa Honduras (No! That's not Mexico!), and he's currently living in Guatemala City (Nope! Not Mexico either. Look for them in a map). His name is Eduardo Corpeño and he bought his Ti-89 somewhere around September 2000. He discovered ticalc.org by January 2001.

He's a good Electronics and Computer Engineer (good at circuit design, programming, pcb manufacturing, circuit simulating in lovely CircuitMaker, etc), but nobody in this rotten country seems to appreciate that.
In here, everyone thinks of an Electronics Engineer as an appliance-repair technician; and a Computer Engineer as an MSOffice literate person. All you have to do to get a good job is know someone who can hire you, no matter what kind of a lamer you are.

As you may have guessed, the author has been in the sad world of the unemployed for a long time now, and he's currently an unsatisfied employee. Uploading Ti-89 software to lovely ticalc.org is one of his hobbies and a nice escape from reality.

w00t!

History

May.21.2k+2	Ti-PLC uploaded to ticalc.org.
May.20.2k+2	Ti-PLC documentation finished.
May.15.2k+2	Many user-friendship features added.
Apr.30.2k+2	Ti-PLC "finished" (nasty bugs fixed).
Apr.25.2k+2	Linkport enabled. External circuit made. Infinite thanks to Josh Christensen.
Apr.20.2k+2	Ti-PLC development retaken after a black black gap.
Jan.22.2k+2	Drawing of the PLC (and other goodies) included.
Oct.??.2k++	First functional demo finished (Never made it to public release).
Sept.22.2k++	TIGCC programming and research begins.