



Open Architecture Control Software

Using The CONVEYOR TLM Instructions

Because of the variety of uses of the information described in this application note, the users of, and those responsible for applying this information must satisfy themselves as to the acceptability of each application and use of the information. In no event will SoftPLC Corporation be responsible or liable for its use, nor for any infringements of patents or other rights of third parties which may result from its use.

SOFTPLC CORPORATION MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE.

SoftPLC Corporation reserves the right to change product specifications at any time without notice.

No part of this document may be reproduced by any means, nor translated, nor transmitted to any magnetic medium without the written consent of SoftPLC Corporation.

SoftPLC and TOPDOC are registered trademarks of SoftPLC Corporation. SoftWIRES is a trademark of SoftPLC Corporation.

© Copyright 1997-2002 SoftPLC Corporation
ALL RIGHTS RESERVED

First Revision: April, 1997
Latest Revision: July, 2002

SoftPLC Corporation
25603 Red Brangus Drive
Spicewood, Texas 78669 USA
Telephone: 512/264-8390 or 1-800-SoftPLC
Fax: 512/264-8399
WWW: <http://www.softplc.com>
Email: info@softplc.com

TABLE OF CONTENTS

<u>SECTION</u>	<u>PAGE</u>
1. INTRODUCTION	1
2. OVERVIEW	1
3. INSTALLING THE TLM	2
4. THE TLI's	3
4.1 FIFOFILL	3
4.2 FIFOSHIFT	4
4.3 FIFOPEEK	5
4.4 FIFOPUT	5

1. INTRODUCTION

SoftPLC Corporation's SoftPLC product employs a unique technology which lets a 'C', C++ or Java programmer add new ladder logic instructions to the instruction set. These loadable instructions are called TLI's. TOPDOC, the ladder logic programming package which supports SoftPLC, automatically learns about the new TLI's as it logs into the SoftPLC. Additionally, simple entries may be placed into an ASCII text file to inform TOPDOC about the TLI's so they may be used offline as well. Then the TLI's can even be used with complete access to the offline emulation and debugging capabilities of SoftPLC Corporation's SoftWIRES.

TLI's may be developed by any competent 'C' programmer who has access to the SoftPLC 'C' Programmer's Toolkit or a Java Programmer with the use of the SoftPLC Java Programmer's Toolkit, both products readily available from SoftPLC Corporation. There are a number of Systems Integrators who are SoftPLC Partners who possess the requisite expertise. End users may also have this capability.

This document describes a number of TLI's, all which reside in a SoftPLC Corporation Loadable Module (TLM). The TLM is called **CONVEYOR** and resides in a file with the name CONVEYOR.TLM. This TLM was developed by SoftPLC Corporation.

2. OVERVIEW

Imagine that you have a moving conveyor machine carrying various colored marker pens and as the pens of a particular color reached a certain point of travel along the conveyor you wanted to push the pen off the conveyor into a bin dedicated to holding that particular color of pen. And say you want to do the same thing for several different colors of pens, each with a different destination bin positioned at a different point of travel along the same conveyor. The ladder logic instructions (TLI's) in the CONVEYOR.TLM are **the tools** to handle this kind of control problem very easily.

The CONVEYOR.TLM is a loadable module for SoftPLC which adds 4 ladder logic instructions (TLI's) which are useful for OEM's and manufacturers responsible for controlling information flow on any kind of conveyor system. The real world is typified by boxes or product of various kinds flowing along a conveyor system. It is useful to keep track of what type of product is present at any position on the conveyor.

The 4 TLI's work on FIFO data structures. A FIFO is a first in first out structure that has an input end and an output end. Data shifts into the input end and comes out the output end in the same order that it was input. FIFO's can be categorized into "variable fullness" or "fixed fullness". The FIFO category used by the 4 conveyor TLI's is fixed fullness. This means for example that if the FIFO has room for 4000 data locations, then there are always 4000 data locations available for use, and the notion of "emptiness" does not apply. This model fits better than the variable fullness FIFO because the size of the conveyor does not change. There are always the same number of physical positions on the conveyor and these match the data locations within the fixed fullness FIFO.

The conveyor is modeled by breaking it down into a number of equal sized physical zones which are identified by an offset from the beginning of the conveyor. They are equal sized, and this size is measured in the direction of travel along the conveyor. As the conveyor travels, the data in the FIFO travels, in a synchronized fashion. When something first goes onto the beginning of the conveyor, it is necessary to take note of its properties of interest. Then this data is immediately stuffed into the input end of the FIFO and will shift through the FIFO as the conveyor travels. At any time it is a simple matter to peak into the FIFO to see data at any offset, and depending on which offset into the FIFO we peek at, this will give information about what is also on the conveyor at that same zone offset.

The TLI's defined here are:

- FIFOFILL Fills all data locations with a given value. This is normally used to zero out the FIFO contents.

- FIFOSHIFT Shifts the contents of the FIFO by a given number of positions and adds a given value to the input end of the FIFO. This is used to keep the FIFO synchronized with the conveyor.

- FIFOPEEK Gets the 16 bit word value at a given offset into the FIFO. This tells you what exists at a given point of travel on the conveyor.

- FIFOPUT Puts a given 16 bit word value at a given offset into the FIFO. This can be used to modify product attributes as the product travels down the conveyor.

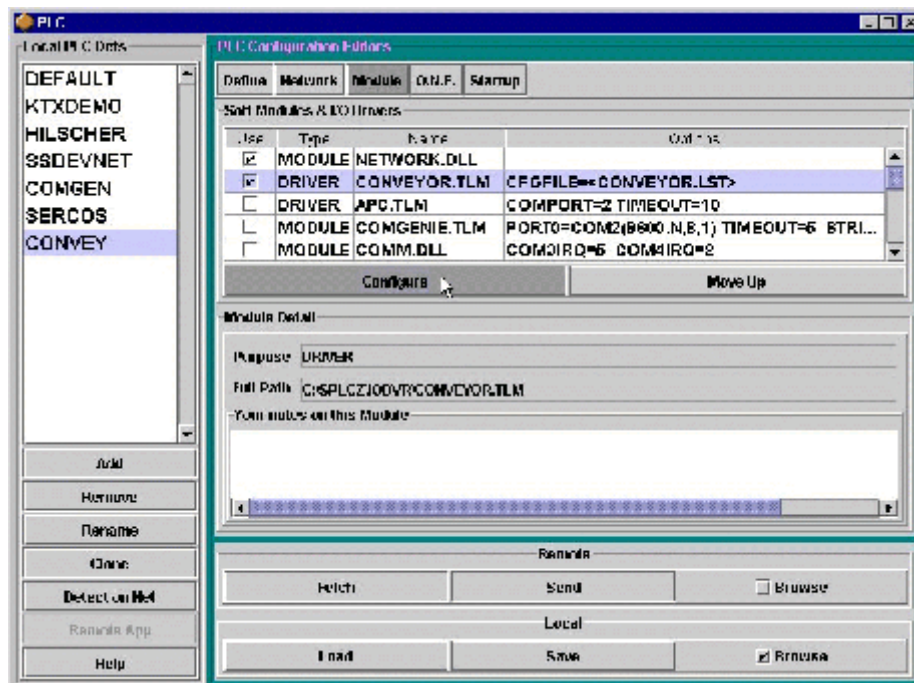
All the TLI's take a parameter called a FIFO ID. The FIFO ID is a number which specifies the logical FIFO which is maintained for you in TLM memory. There is no limit to the number of FIFO's supported, nor the size of each FIFO, except for the limits imposed by available physical memory.

Each FIFO consists of 16 bit wide words. If you need more than 16 bits per data location, then you can operate more than one FIFO in parallel.

3. INSTALLING THE TLM

To install the CONVEYOR TLM, you must place the TLM file and the sample CONVEYOR.LST file into the \SPLCZIODVR directory on the SoftPLC CPU (normally done via an FTP transfer). Then you must create a .def file within your \SoftPLC\plc directory on your development system (a sample.def file is provided with your install).

Using TOPDOC NexGen's PLC Configuration Editor, go to the Module tab and select the CONVEYOR.TLM for use.



The **CONVEYOR** TLM can accept configuration option parameters. One configuration option is mandatory.

```
CFGFILE=<CONVEYOR.LST>
```

CONVEYOR.LST file is an ASCII text file that simply contains the sizes of each FIFO that you want to use. Each size is on a separate line. The first one encountered has a FIFO ID of 0, the second one has a FIFO ID of 1, etc. Blank lines are ignored, and anything to the right of a ';' (semi-colon) is also ignored. For example:

```
4000          ; FIFO 0
30000        ; FIFO 1
```

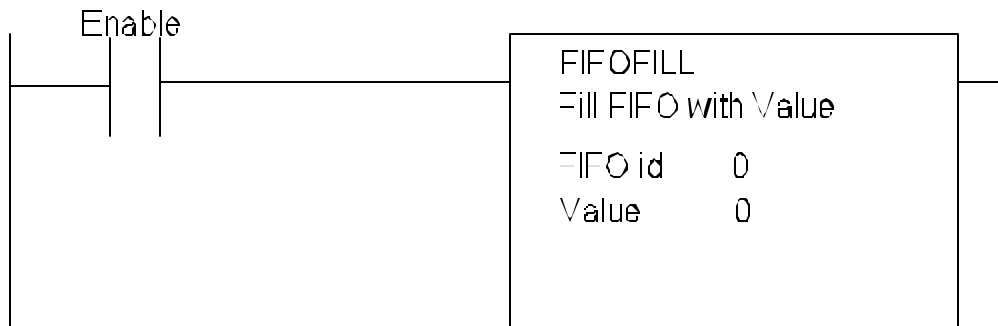
This CONVEYOR.LST file will cause FIFO ID 0 and FIFO ID 1 to be created. FIFO 0 will be 4000 elements in size, and FIFO 1 will be 30,000 elements in size.

From TOPDOC NexGen's Module Editor, while positioned on the selected CONVEYOR.TLM line you can click Configure to FETCH the sample CONVEYOR.LST from your PLC to edit the parameters.

4. THE TLI's

4.1 FIFOFILL

This TLI fills all data locations within a FIFO to with a given value. This is normally used to zero out the FIFO contents.



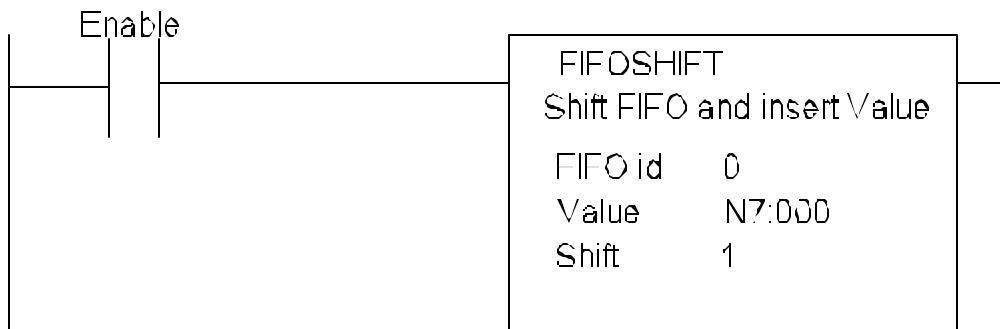
The instruction works whenever the rung condition is true. You may use a one shot instruction (ONS) in series with, and to the right of, the enable conditions to obtain rising edge triggered behavior.

At startup of SoftPLC all FIFO's are filled with zeros. It is often desirable to fill a FIFO with a known value at any transition to a RUN mode. This can be accomplished by using the first scan bit in the STATUS file as the Enable condition (S1/15), as this bit is on for exactly one scan every time there is a transition from a program mode to a run mode.

4.2 FIFOSHIFT

This TLI shifts the contents of the FIFO data either forward or backward a number of positions. The SHIFT parameter indicates both the direction and amount of the shift. Hence it is possible to shift more than one position with a single invocation of this instruction. If the sign of SHIFT is positive, then the INSERT parameter's value is inserted onto the insert end of the FIFO for a count of SHIFT times. If the sign of the SHIFT is negative, then the INSERT parameter's value is inserted on the output end of the FIFO for a count of SHIFT times. Negative SHIFT values are supported in case you want to track backward movement of a conveyor. In the course of shifting, it is necessary to discard the same number of values as are inserted. These values are irretrievably lost. If you need these values, you should obtain them with the FIFOPEEK instruction before calling this FIFOSHIFT instruction.

The most common usage is to insert a product attribute with a SHIFT value of one. This places the new 16 bit word attribute into the FIFO at position 0 (zero). Another frequent usage is to shift in "empty slots" which are used to indicate no product at those positions. Here the SHIFT amount may be larger than 1 (one) indicating that no product is present on the conveyor at those positions. Often a value of 0 is used as the INSERT value symbolizing "no product".



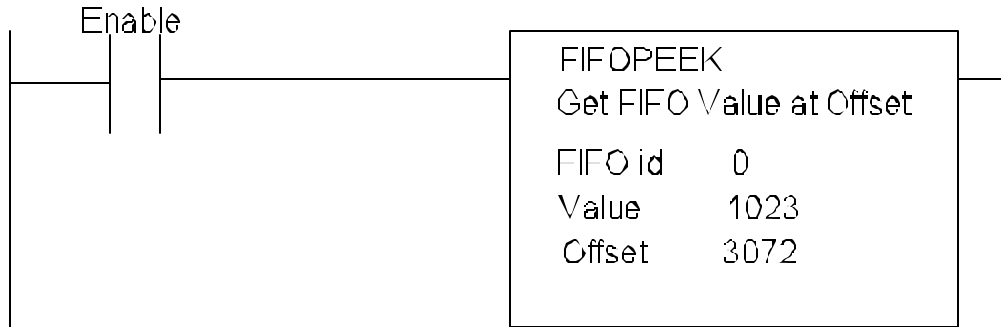
The instruction works whenever the rung condition is true. You may use a one shot instruction (ONS) in series with, and to the right of, the enable conditions to obtain rising edge triggered behavior.

If you want to track the progress of the FIFO, the number of times it has been shifted since being FIFOFILL-ed with zeros, lets say, then you can put a count up (CTU) counter instruction in a parallel output branch so that you have a count of how many times the FIFOSHIFT has been energized. Providing the following 3 conditions are met, the counter accumulated value will track the internal shift pointer within the FIFO:

- 1) The Shift value is 1 (You would use a CTD if -1 is the Shift).
- 2) You are using a One-Shot instruction (ONS) in front of the FIFOSHIFT. This is necessary because the CTU instruction is rising edge triggered, whereas the FIFO instructions are rung true triggered.
- 3) When the counter becomes equal to the size of the FIFO, you reset it to zero.

4.3 FIFOPEEK

This TLI gets the contents of the 16 bit FIFO data word at the given offset from the insert end. The first position is numbered zero, the second position of travel is numbered 1, etc. Any use of the FIFOSHIFT with a positive Shift parameter will put the Value parameter into Offset 0. So if you were to immediate FIFOPEEK at offset zero, you would see exactly what you just shifted in.



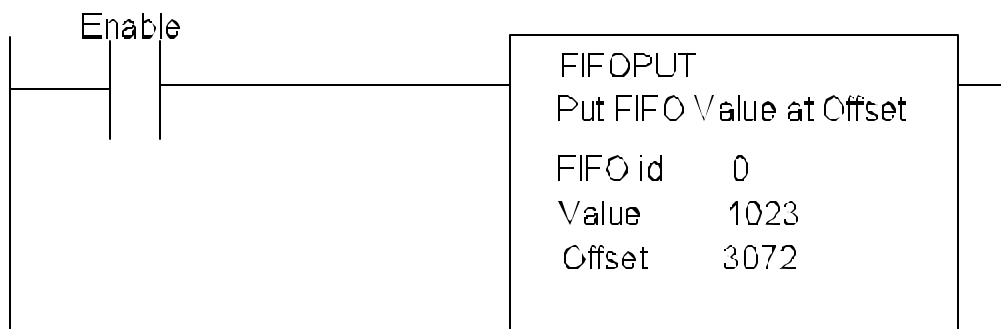
The instruction works whenever the rung condition is true. You may use a one shot instruction (ONS) in series with, and to the right of, the enable conditions to obtain rising edge triggered behavior.

Use this instruction with a known Offset parameter to watch the product attributes at a particular physical point on the conveyor. You can test the Value parameter retrieved with an Equal (EQU) instruction to know when there is a product of interest at that position. When the EQU instruction is true, you can energize an output to push the product off the conveyor at that point.

With only a few rungs it is possible to handle very fast, sophisticated sorting systems.

4.4 FIFOPUT

This TLI puts the contents of a given 16 bit word VALUE into the FIFO at a given OFFSET from the insert end. The first offset is numbered zero, the second offset is numbered 1, etc. This instruction is used to modify product attributes as the product is modified traveling down the conveyor. If you are using bit mapped product attributes, you might have to first FIFOPEEK at an offset, then set or reset bits in this copy, and then finally FIFOPUT the new value back at the same OFFSET.



The instruction works whenever the rung condition is true. You may use a one shot instruction (ONS) in series with, and to the right of, the enable conditions to obtain rising edge triggered behavior.