Cookie Factory Virtual Model Documentation: OpenPLC Program

Contents

[Introduction 1](#_Toc199773627)

[Basic Program Description and Opening Process 1](#_Toc199773628)

[Description 1](#_Toc199773629)

[Opening the Program 1](#_Toc199773630)

[Function Blocks 4](#_Toc199773631)

[What is a Function Block? 4](#_Toc199773632)

[Function Blocks in This Program 4](#_Toc199773633)

[real\_to\_words 4](#_Toc199773634)

[words\_to\_real 5](#_Toc199773635)

[flour\_alarm\_check 5](#_Toc199773636)

[sugar\_alarm\_check 5](#_Toc199773637)

[hopper\_alarm\_check 5](#_Toc199773638)

[vacuum\_alarm\_rpm 5](#_Toc199773639)

[freq\_function 6](#_Toc199773640)

[Tags 7](#_Toc199773641)

Introduction

This document is to assist you in modifying/using the OpenPLC program created for the Cookie Factory Virtual Model. It is recommended that you first understand the OpenPLC documentation, for both editor and runtime, before following the steps outlined below.

The following document contains information on the tags used in the program, the different function blocks created, and their structure. It also describes how to export the program to be used in OpenPLC runtime. A guide on how to install OpenPLC editor and runtime can be found at the following link: <https://autonomylogic.com/docs/openplc-overview/>

Note that this document is in no way complete, because the program itself is not complete. So check back in the box folder for any updates to the program and its documentation, though I will probably let you know when an update occurs.

# Basic Program Description and Opening Process

## Description

The purpose of this program is to act as the controller between the python simulation, and the human machine interface (HMI) running on Ignition Vision. This is to emulate a somewhat realistic set up that you would see in a factory or manufacturing plant. This program is not finished, so there are some debugging tags being used, but it should be somewhat of a realistic representation of a PLC in a manufacturing environment based on it’s tags and the values they represent.

## Opening the Program

After installation of the OpenPLC editor, you will be able to start it from the start menu on your Windows machine. Once OpenPLC editor is open you should see the following window:

A screenshot of a computer

AI-generated content may be incorrect.

Currently no project is loaded, to load the PLC program used for the cookie factory, download the folder named Cookie-Manufacturing-Model-PLC\_Project from the parent folder of this document.

Now open the downloaded folder by navigating to File->Open and selecting the folder.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

After opening the program, you should see a tree populated in the Project pane. Click the box to the left of the “Function Blocks” item to expand its tree, allowing you to see each function block, double clicking on a function block will show their variables and the code that they’re made up of, which will either be C or Structured Text (ST).

A screenshot of a computer

AI-generated content may be incorrect.

You can double click the item labeled “tag\_display” to show all the tags used in the program as well the function block diagram. This diagram acts as the brains of the program that bring everything together.

A screenshot of a computer

AI-generated content may be incorrect.

The function block diagram may not make a whole lot of sense now, but we’ll dive into function block and tag so this diagram will make a whole lot more sense.

# Function Blocks

## What is a Function Block?

A function block represents a reusable block of code that encapsulates a specific function, it can have any number of inputs and any number of outputs. Function blocks are primarily used to create relationships between different tags via a function block diagram.

## Function Blocks in This Program

There are a total of seven custom function blocks in this PLC program, each having their own unique function, as well as one basic function block included in OpenPLC’s library (ADD).

### real\_to\_words

This function takes a real 32-bit float as an input and then returns two 16-bit words that correlate to the lower and upper halves of the float. This is because Ignition doesn’t allow addressing floating point numbers explicitly. So, our program must break each float up into two 16-bit halves to be transferred to Ignition. Once the halves reach our HMI, they are then stitched back together to create a floating-point number.

### words\_to\_real

This function is essentially the reverse of real\_to\_words. So, it has two 16-bit word inputs and produces a 32 bit real number.

### flour\_alarm\_check

This function checks to see If the total flour weight is less than 1000 pounds, if it is then it returns true. This function has a 32-bit real number, representing the amount of flour in the flour silo, as an input and a Boolean, representing if the alarm is active or not, as an output.

### sugar\_alarm\_check

This function checks to see If the total sugar weight is less than 1000 pounds, if it is then it returns true. This function has a 32-bit real number, representing the amount of sugar in the sugar silo, as an input and a Boolean, representing if the alarm is active or not, as an output.

### hopper\_alarm\_check

This function checks to see If the total hopper weight is less than 100 pounds, if it is then it returns true. This function has a 32-bit real number, representing the amount of material in the hopper, as an input and a Boolean, representing if the alarm is active or not, as an output.

### vacuum\_alarm\_rpm

This function checks to see if the vacuum is running at too slow of a speed for the material being conveyed. This function has two inputs, one is Dv which is a Boolean representing the material being conveyed, the other is a word variable representing the vacuum’s current RPM.

Using these two inputs and a best curve function created using some sample data, the function calculates the current cubic feet per minute (CFM) of material that is being moved. This value is compared with the two constant threshold values for sugar and flour respectively. If they are lower than the threshold the alarm will ring. This function has two outputs, one being a 32-bit real number representing the CFM, and the other being a Boolean that represents whether the alarm is active or not.

The data used for the best curve function can be found on the following datasheet:

<https://www.pdblowers.com/wp-content/uploads/2008/03/Roots-RAM-DVJ-406J-412J-616J-spec_RT-VAC-EXHAUST-RAM-DVJ-WHISPAIR-406-616_12-23.pdf>

These rows/columns in particular:  
A table with numbers and numbers

AI-generated content may be incorrect.

I used this datasheet because it has high enough CFM for both ingredients and the relationship between RPM and CFM is completely dependent on the motor and

blower/vacuum being used. The vacuum represented in this virtual model most closely resembles the one shown in the datasheet above, specifically the 616 DVJ.

The CFM requirements for flour and sugar were gathered from A. Bhatia’s course on Pneumatic Conveying Systems, which can be found in the papers folder under the main 5.1 folder in Box.

### freq\_function

This function determines the frequency at which the motor within the vacuum operates, and whether the vacuum should be operated in manual or automatic mode. The frequency of the motor depends on the material being conveyed if the automatic mode is selected in the HMI. If the manual mode is selected on the HMI, then the operator can change the speed of the vacuum as they please.

This function has three inputs, and one output. Two of the inputs are Booleans, one of which represents whether the vacuum is in manual control mode the other representing the state of the diverter valve. The last input represents the input frequency as a 16-bit word.

The output frequency is a 16-bit word that represents the updated frequency the vacuum will be operating at, if the diverter valve is switched to flour, it will operate at a frequency of 23, if the diverter valve is switched to sugar, it will operate at a frequency of 33, and if the manual Boolean is set to true, then the output frequency will equal the input frequency.

# Tags

There are a total of 48 separate tags. See the table below for their values and relative information. Note that some tags have an OpenPLC address and a Modbus address, this is because OpenPLC uses it’s own addressing system which is outlined here:

<https://autonomylogic.com/docs/2-5-modbus-addressing/>

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | **Type** | **OpenPLC address** | **Modbus address** | **Initial Value** | **Note** |
| Ics\_1\_low | WORD(16 bits) | %QW100 | 100 |  | Load cell 1 for sugar (lower int) |
| lcs\_1\_high | WORD(16 bits) | %QW101 | 101 |  | Load cell 1 for sugar (higher int) |
| Ics\_2\_low | WORD(16 bits) | %QW102 | 102 |  | Load cell 2 for sugar (lower int) |
| lcs\_2\_high | WORD(16 bits) | %QW103 | 103 |  | Load cell 2 for sugar (higher int) |
| Ics\_3\_low | WORD(16 bits) | %QW104 | 104 |  | Load cell 3 for sugar (lower int) |
| lcs\_3\_high | WORD(16 bits) | %QW105 | 105 |  | Load cell 3 for sugar (higher int) |
| Ics\_4\_low | WORD(16 bits) | %QW106 | 106 |  | Load cell 4 for sugar (lower int) |
| lcs\_4\_high | WORD(16 bits) | %QW107 | 107 |  | Load cell 4 for sugar (higher int) |
| lcf\_1\_low | WORD(16 bits) | %QW108 | 108 |  | Load cell 1 for flour (lower int) |
| lcf\_1\_high | WORD(16 bits) | %QW109 | 109 |  | Load cell 1 for flour (higher int) |
| lcf\_2\_low | WORD(16 bits) | %QW110 | 110 |  | Load cell 2 for flour (lower int) |
| lcf\_2\_high | WORD(16 bits) | %QW111 | 111 |  | Load cell 2 for flour (higher int) |
| lcf\_3\_low | WORD(16 bits) | %QW112 | 112 |  | Load cell 3 for flour (lower int) |
| lcf\_3\_high | WORD(16 bits) | %QW113 | 113 |  | Load cell 3 for flour (higher int) |
| lcf\_4\_low | WORD(16 bits) | %QW114 | 114 |  | Load cell 4 for flour (lower int) |
| lcf\_4\_high | WORD(16 bits) | %QW115 | 115 |  | Load cell 4 for flour (higher int) |
| lch\_low | WORD(16 bits) | %QW98 | 98 |  | Load cell for hopper (higher int) |
| lch\_high | WORD(16 bits) | %QW99 | 99 |  | Load cell for hopper (lower int) |
| vacuum\_freq | WORD(16 bits) | %QW1000 | 1000 | 0 | Vacuum frequency |
| vacuum\_rpm | WORD(16 bits) | %QW1001 | 1001 |  | Vacuum RPM |
| mixer\_freq | WORD(16 bits) | %QW1002 | 1002 |  | Mixer frequency (not implemented) |
| mixer\_rpm | WORD(16 bits) | %QW1003 | 1003 |  | Mixer RPM (not implemented) |
| lcs\_1 | REAL(32 bits) | %MD0 |  |  | Load cell 1 for sugar (combined float) |
| lcs\_2 | REAL(32 bits) | %MD1 |  |  | Load cell 2 for sugar (combined float) |
| lcs\_3 | REAL(32 bits) | %MD2 |  |  | Load cell 3 for sugar (combined float) |
| lcs\_4 | REAL(32 bits) | %MD3 |  |  | Load cell 4 for sugar (combined float) |
| lcf\_1 | REAL(32 bits) | %MD4 |  |  | Load cell 1 for flour (combined float) |
| lcf\_2 | REAL(32 bits) | %MD5 |  |  | Load cell 2 for flour (combined float) |
| lcf\_3 | REAL(32 bits) | %MD6 |  |  | Load cell 3 for flour (combined float) |
| lcf\_4 | REAL(32 bits) | %MD7 |  |  | Load cell 4 for flour (combined float) |
| lch | REAL(32 bits) | %MD8 |  | 0 | Load cell for hopper (combined float) |
| flour\_weight | REAL(32 bits) | %MD9 |  | 0 | Flour weight (combined float) |
| flour\_weight\_low | WORD(16 bits) | %QW116 | 116 |  | Flour weight (lower integer) |
| flour\_weight\_high | WORD(16 bits) | %QW117 | 117 |  | Flour weight (higher integer) |
| sugar\_weight | REAL(32 bits) | %MD10 |  | 0 | Sugar weight (combined float) |
| sugar\_weight\_low | WORD(16 bits) | %QW218 | 218 |  | Sugar weight (lower integer) |
| sguar\_weight\_high | WORD(16 bits) | %QW219 | 219 |  | Sugar weight (higher integer) |
| rv\_1 | BOOL | %QX87.4 | 700 |  | Rotary valve 1 (flour) |
| rv\_2 | BOOL | %QX87.5 | 701 |  | Rotary valve 2 (sugar) |
| rv\_3 | BOOL | %QX87.6 | 702 |  | Rotary valve 3 (hopper) |
| dv | BOOL | %QX87.7 | 703 |  | Diverter valve |
| vacuum | BOOL | %QX88.0 | 704 |  | Vacuum on state |
| mixer | BOOL | %QX88.1 | 705 |  | Mixer on state (not implemented) |
| flour\_alarm | BOOL | %QX88.2 | 706 | FALSE | Flour alarm (if flour\_weight<1000) |
| sugar\_alarm | BOOL | %QX88.3 | 707 | FALSE | Sugar alarm (if sugar\_weight<1000) |
| hopper\_alarm | BOOL | %QX88.4 | 708 | FALSE | Hopper alarm (if lch<100) |
| vacuum\_alarm | BOOL | %QX88.5 | 709 | FALSE | Vacuum alarm |
| vacuum\_manual | BOOL | %QX88.6 | 710 | FALSE | Vacuum manual control state |
| CFM | REAL(32 bits) | %MD11 |  |  | Calculated cubic feet per min of material |

The orange fields of the table represent tags used by the HMI in Ignition, the blue fields of the table represent tags used by the python simulation, and the green fields of the table represent tags that are used by both the python simulation and Ignition. The gray fields are used only by the PLC program, some of which are used for OpenPLC Runtime’s monitoring feature.