**МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»**

**(РУТ (МИИТ)**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Институт транспортной техники и систем управления**

**Кафедра «Управление и защита информации»**

**ОТЧЕТ**

**по теме «Разработка подпрограмм в National Instruments LabView»**

**по практике  
Вариант ­‑ 14**

|  |  |  |
| --- | --- | --- |
|  |  | Выполнил:  Студент группы ТКИ-241  Небоваренков С.К.  Проверил:  Сафронов Антон Игоревич |

**Москва – 2025**

[1. Задание № 1 3](#_Toc200810351)

[1.1. Цель работы: 3](#_Toc200810352)

[1.2. Индивидуальная часть: 3](#_Toc200810353)

[1.3. Условие 3](#_Toc200810354)

[1.4. Программа: 4](#_Toc200810355)

[1.5. Блок-схема 5](#_Toc200810356)

[1.6. Проверка результатов 7](#_Toc200810357)

[1.7. Вывод: 9](#_Toc200810358)

[2. Задание № 2 10](#_Toc200810359)

[2.1. Цель работы: 10](#_Toc200810360)

[2.2. Подготовка исходных данных для построения графиков функциональных зависимостей: 10](#_Toc200810361)

[2.3. Условие задания: 10](#_Toc200810362)

[2.4. Программа: 11](#_Toc200810363)

[2.5. Проверка результатов: 15](#_Toc200810364)

[2.6. Блок-схема: 18](#_Toc200810365)

[2.7. Вывод: 20](#_Toc200810366)

[3. Задание № 3 21](#_Toc200810367)

[3.1. Цель работы: 21](#_Toc200810368)

[3.2. Условие задания: 21](#_Toc200810369)

[3.3. Программа: 21](#_Toc200810370)

[3.4. Проверка результатов: 26](#_Toc200810371)

[3.5. Блок-схема: 28](#_Toc200810372)

[3.6. Вывод: 31](#_Toc200810373)

[4. Задание № 4 31](#_Toc200810374)

[4.1. Цель работы: 31](#_Toc200810375)

[4.2. Условие задания: 31](#_Toc200810376)

[4.3. Программа: 31](#_Toc200810377)

[4.4. Результат программы: 48](#_Toc200810378)

[4.5. Блок-схема: 58](#_Toc200810379)

[4.6. Вывод: 58](#_Toc200810380)

1. Задание № 1
   1. **Цель работы:**

Освоение навыков разработки пользовательских процедур. Закрепление раздела «Ряды» из курса высшей математики. Закрепление раздела «Функции и процедуры» из курса алгоритмизации и технологий программирования.

* 1. Индивидуальная часть:

Разработать пользовательскую подпрограмму (виртуальный прибор), реализующую вычисление с заданной точностью указанных по варианту функций. В случае образования функции факториала при разложении указанной функции в ряд использовать подпрограмму, составленную в общей части данного задания. По итогам создания и отладки пользовательской подпрограммы создать виртуальный прибор, блок-диаграмма которого содержала бы не более чем:

- элементы ввода данных,

- элементы вывода данных,

- пользовательскую подпрограмму.

* 1. **Условие**

В пакете прикладных программ *National Instruments LabView* разработать подпрограмму (виртуальный прибор) для расчёта/исследования функции факториала. Определить границы корректного вычисления факториала и ограничить диапазон пользовательского ввода входного значения. , вычисленный с учётом *N* членов ряда (циклическая структура *For*).

* 1. Программа:

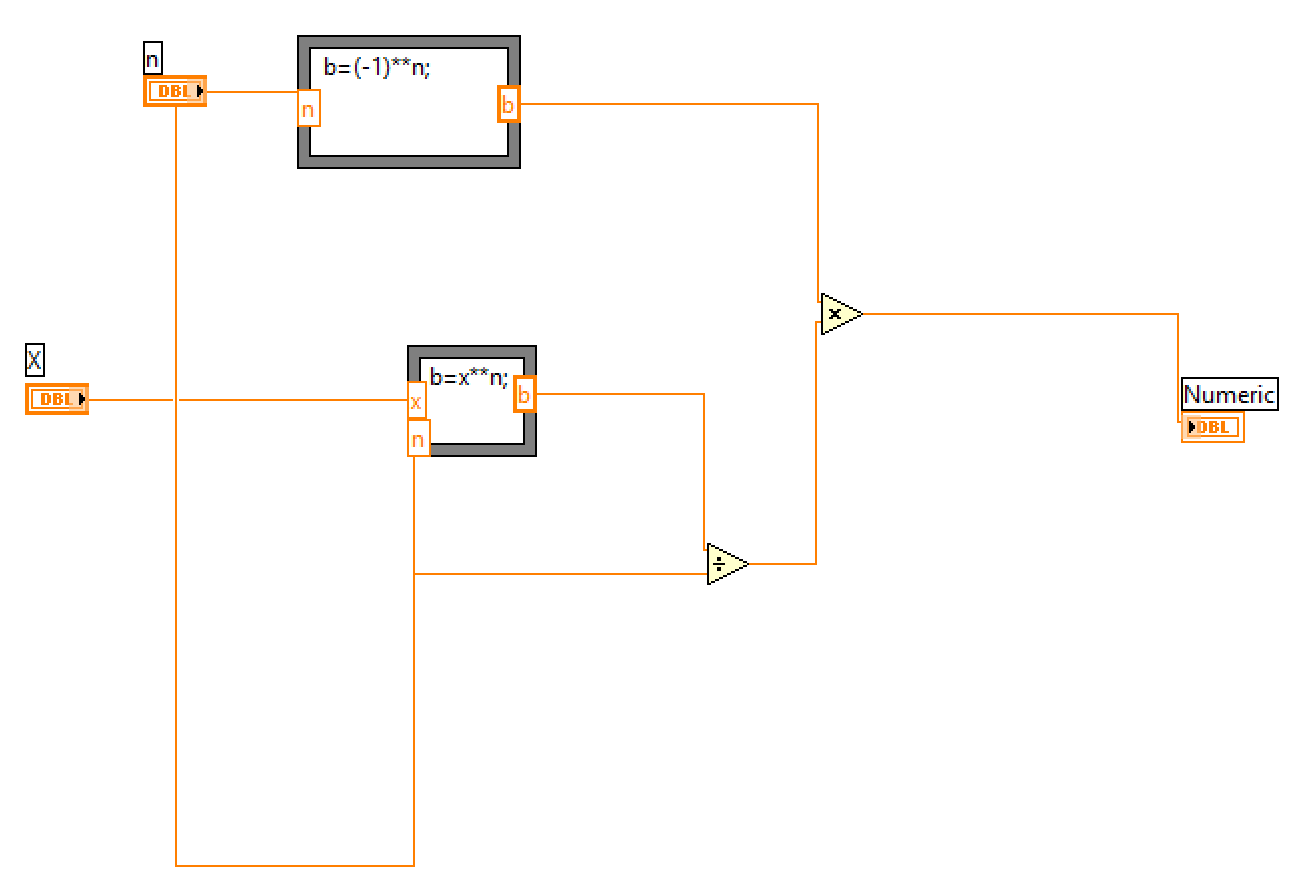


Рисунок. 1 – *first\_part.vi*

В данной части программы происходит расчет функции , результат данного выражение будет использован в дальнейшем для расчета суммы членов ряда.

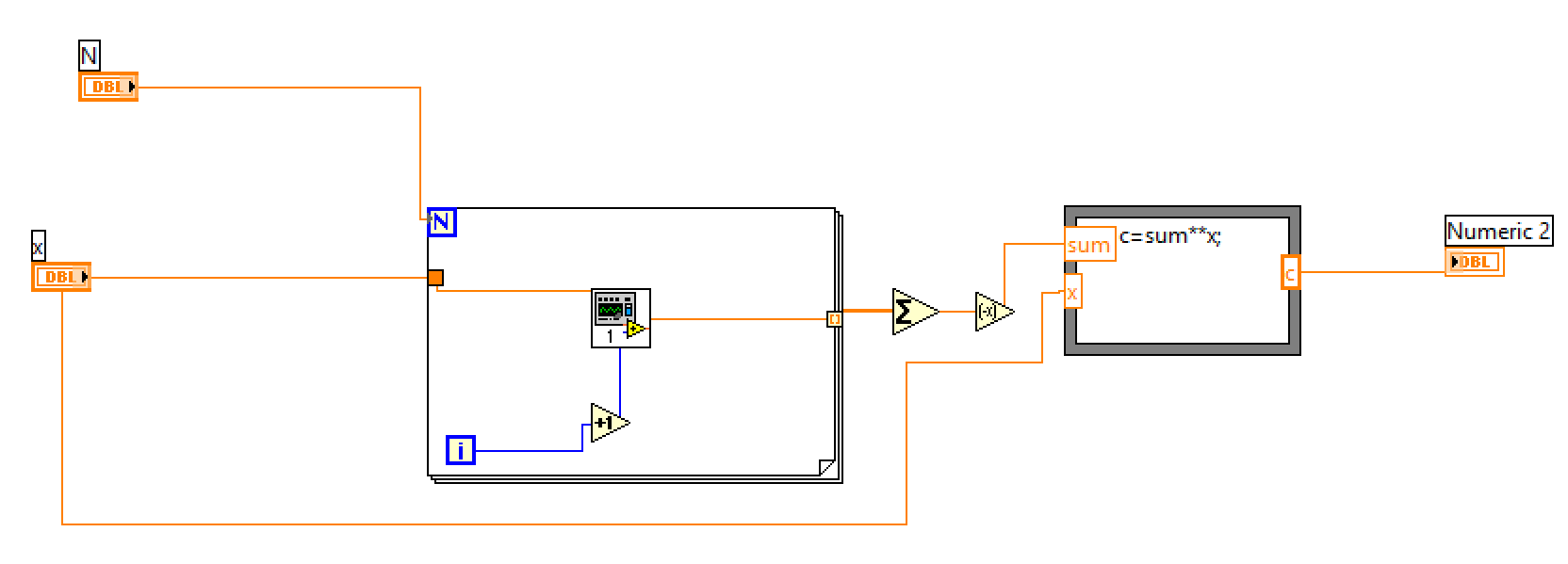


Рисунок. 2 – *last\_part.vi*

Программа использует введенные параметры x и N, передавая их в цикл для расчета суммы членов ряда.

* 1. Блок-схема

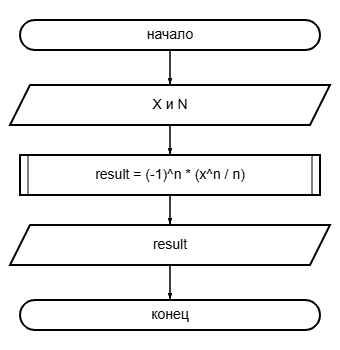


Рисунок. 3 – блок-схема для функции *first\_part.vi*

На данной блок схеме описан алгоритм программы *first\_part.vi.* Передаются значения x и N, затем программа рассчитывает результат и передает его на вывод.

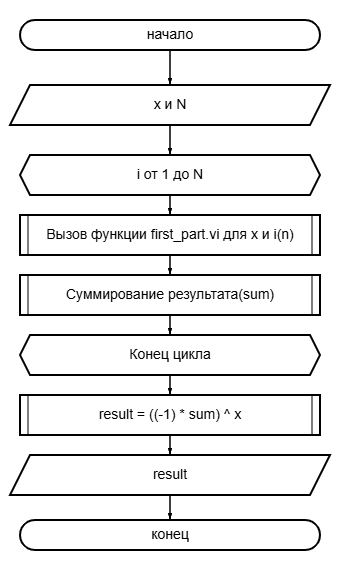


Рисунок. 4 – блок-схема для функции *last\_part.vi*

На данной блок-схеме описан процесс работы программы *last\_part.vi.* Сначала передаются параметры x и N. Затем в цикле от 1 до N вызывается функция first\_part.vi(x, i), где i = n. Происходит суммирование полученных значений. Затем сумма умножается на -1 и возводится в степень x. Этот результат и передается в ответ.

* 1. Проверка результатов

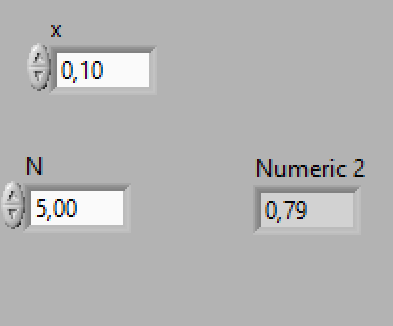


Рисунок. 5 – результат выполнения программы при 0.1 *LabView*

На данном рисунке представлен результат вывода программы для 5 членов ряда и значении x = 0.1 в программе *LabView*.

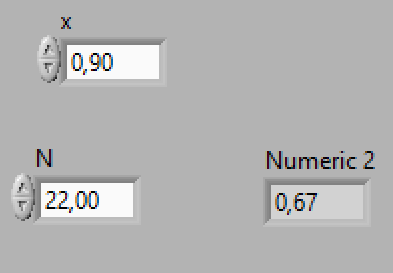


Рисунок. 6 – результат выполнения программы при 0.9 *LabView*

На данном рисунке представлен результат вывода программы для 22 членов ряда и значении x = 0.9 в программе *LabView*.

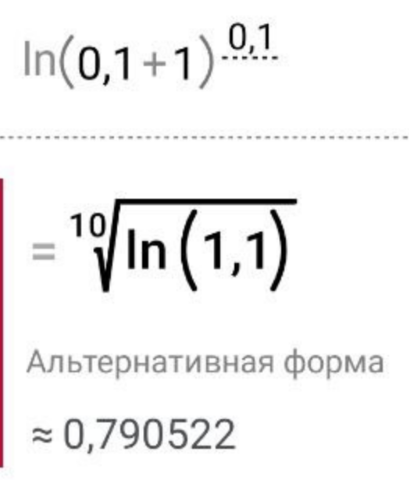


Рисунок. 7 – результат выполнения программы в *Photomath*

На рисунке показан результат проверки вычислений, полученный в программе *Photomath* для значения x = 0.1*.*



Рисунок. 8 – результат выполнения программы в *Photomath*

На рисунке показан результат проверки вычислений, полученный в программе *Photomath* для значения x = 0.9*.*

* 1. Вывод:

С помощью программы *LabView* можно научиться вычислять как простые примеры, так и сложные выражения. А также, можно проверить верность выполненного задания с помощью программы *Photomath*, решив пример в обеих программах и сравнить результат.

1. Задание № 2
   1. Цель работы:

закрепление навыков использования графической структуры цикла *For* (или структуры цикла *While* в зависимости от реализации), приобретение навыков конвертации и проецирования значений периодических и не периодических функций на окружность (преобразования в полярную систему координат).

* 1. Подготовка исходных данных для построения графиков функциональных зависимостей:

Исходными данными для построения графиков функциональных зависимостей являются два параметра:

- диапазон построения ([*Xn*; *Xk*]);

- шаг построения (*h*).

Диапазон построения, в свою очередь, распадается на два параметра:

- начало диапазона построения (*Xn*);

- конец диапазона построения (*Xk*).

* 1. Условие задания:

В пакете прикладных программ *National Instruments LabView* создать виртуальный прибор «Полярные координаты», в рамках которого выполнить построение графика заданной по варианту функции в декартовой системе координат.

* 1. Программа:

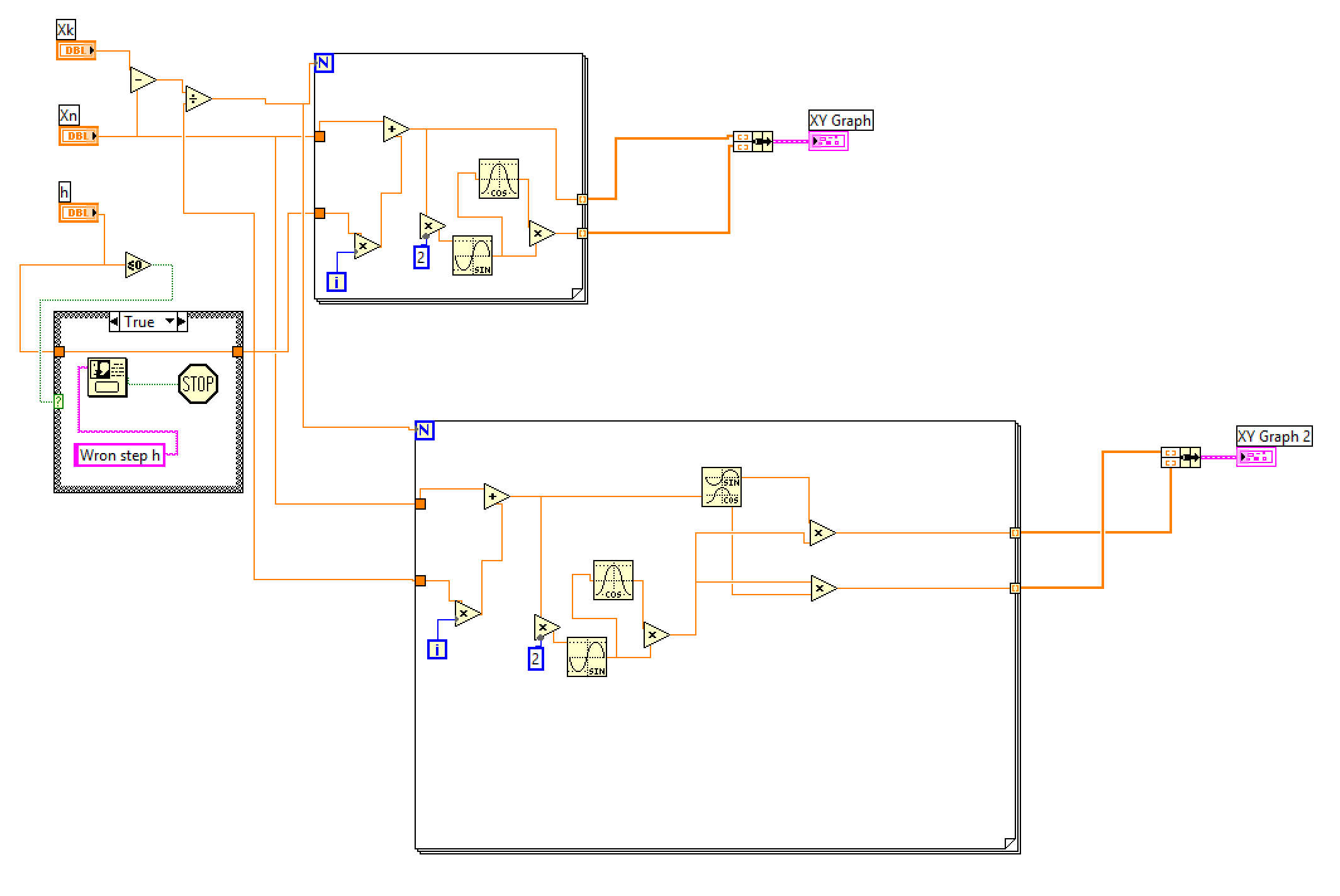


Рисунок. 9 – Программа *TASK2.vi*

На рисунке представлена программа *TASK2.vi* целиком. Далее рассмотрим ее подробнее.

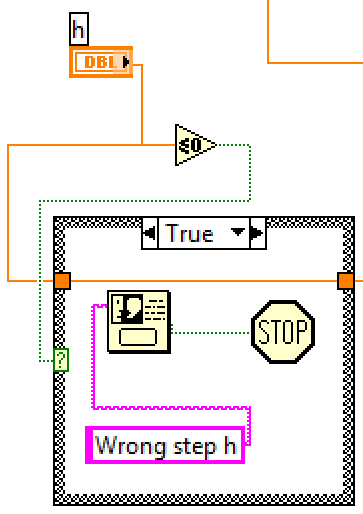


Рисунок. 10 – Программа *TASK2.vi* (проверка шага)

На рисунке представлен алгоритм проверки шага на отрицательность.

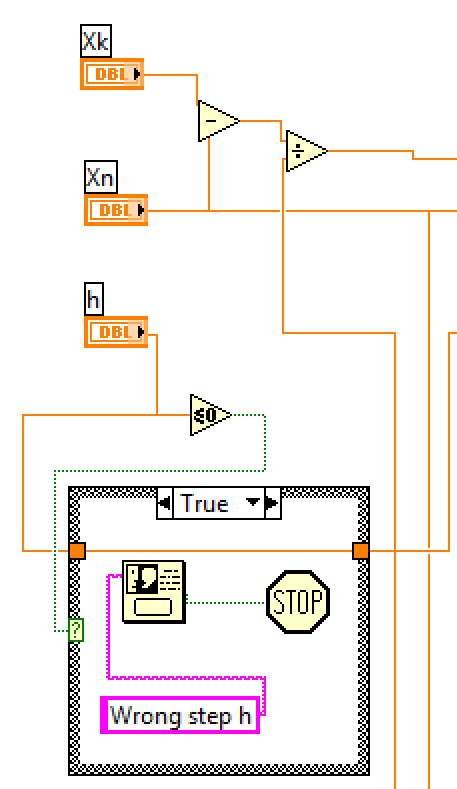


Рисунок. 11 – Программа *TASK2.vi* (Нахождение количества точек)

На рисунке показан алгоритм нахождения точек после проверки шага на отрицательность.

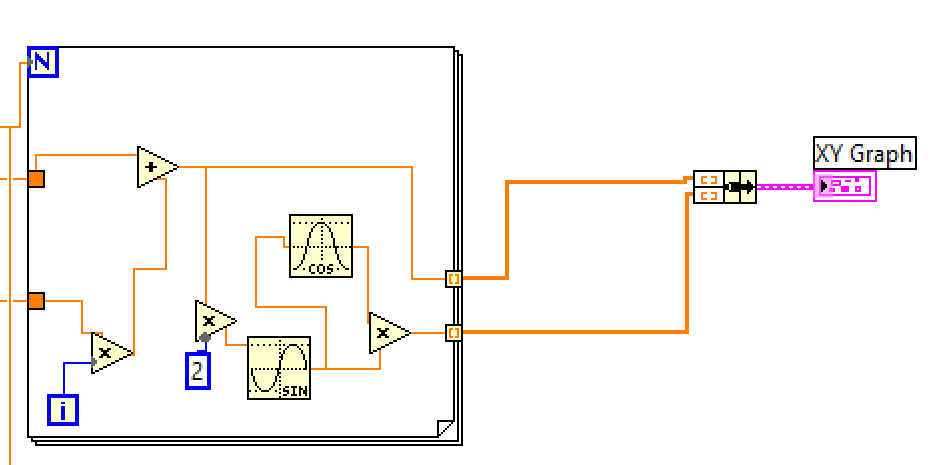


Рисунок. 12 – Программа *TASK2.vi* (Нахождение значений функции и построение графика в декартовой системе координат)

На рисунке представлен цикл, на вход которого подается значение шага, количество полученных точек и начальное значение. Шаг умножается на значение i и прибавляется к начальному значению – это параметр x. Параметр y рассчитывается по формуле Они передаются и преобразуются в кластер, по которому строится график в декартовой системе.

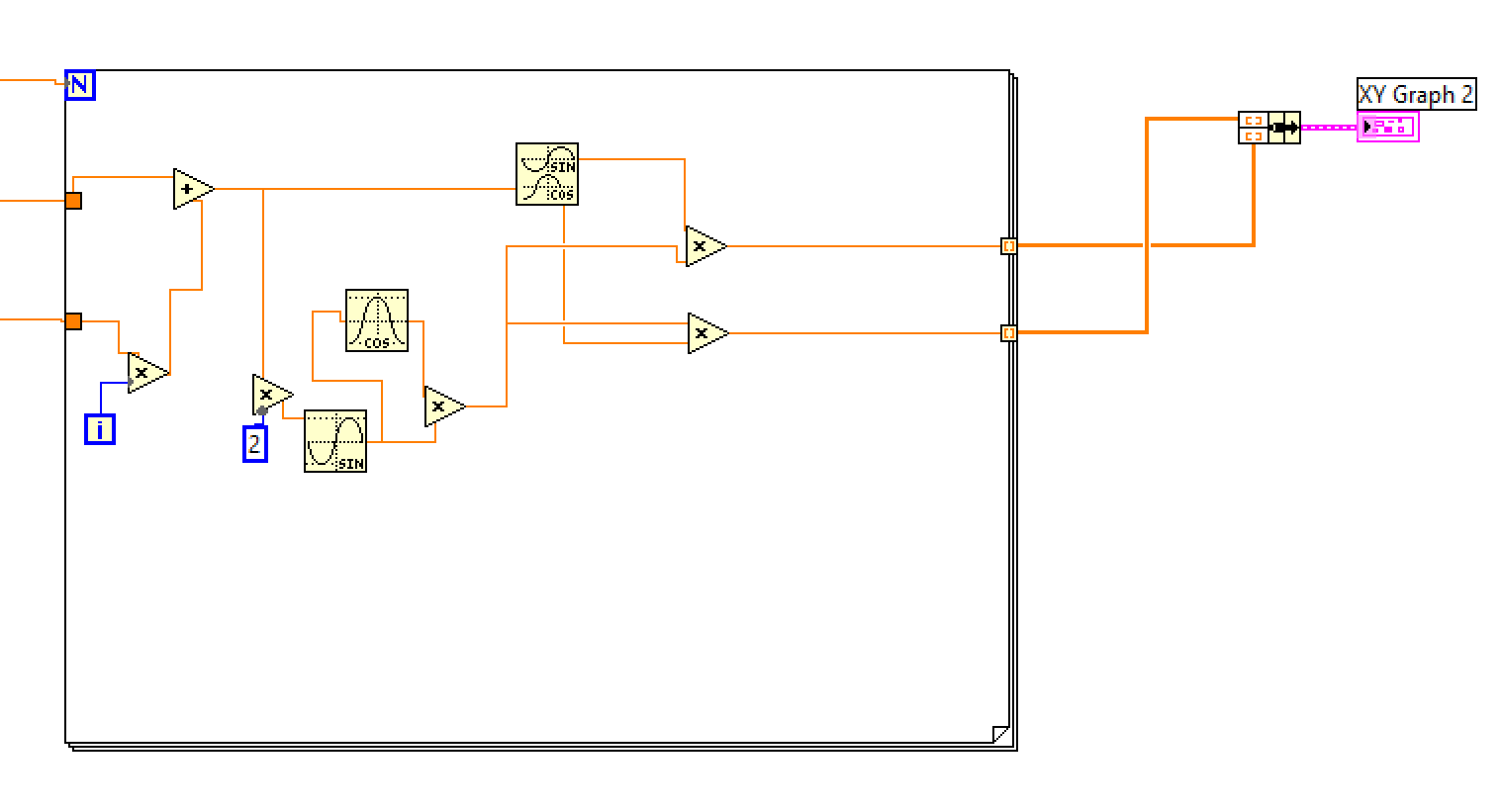


Рисунок. 13 – Программа *TASK2.vi* (Нахождение значений функции и построение графика в полярной системе координат)

На рисунке представлен цикл, на вход которого подается значение шага, количество полученных точек и начальное значение. Шаг умножается на значение i и прибавляется к начальному значению – это параметр φ. Параметр y рассчитывается по формуле Они передаются и преобразуются в кластер, по которому строится график в полярной системе.

* 1. Проверка результатов:

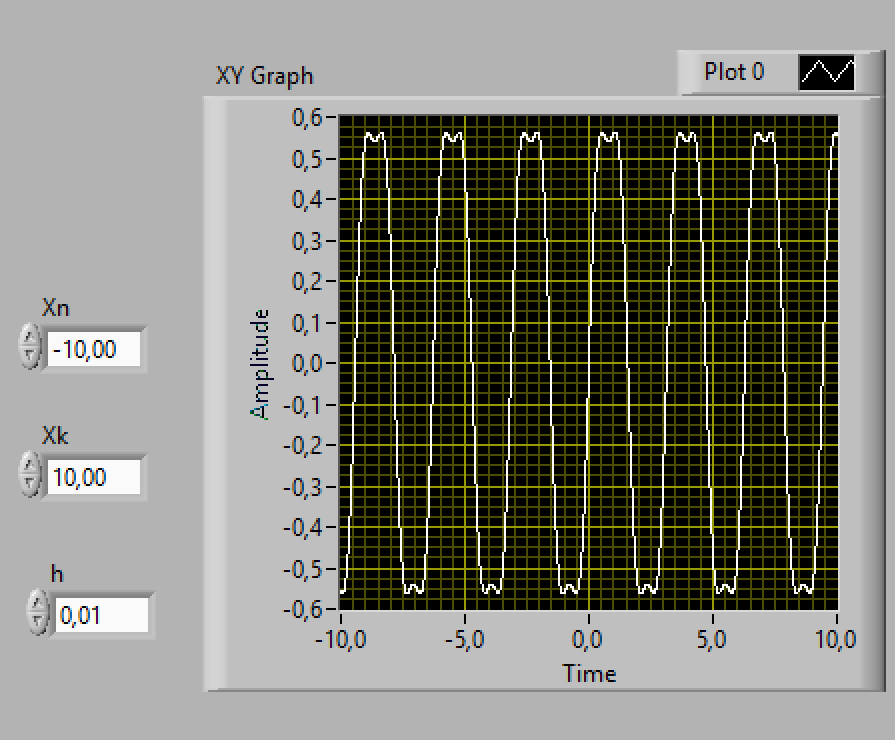


Рисунок. 14 – Результат работы программы в *LabView*

На рисунке результат вывода программы – график функции

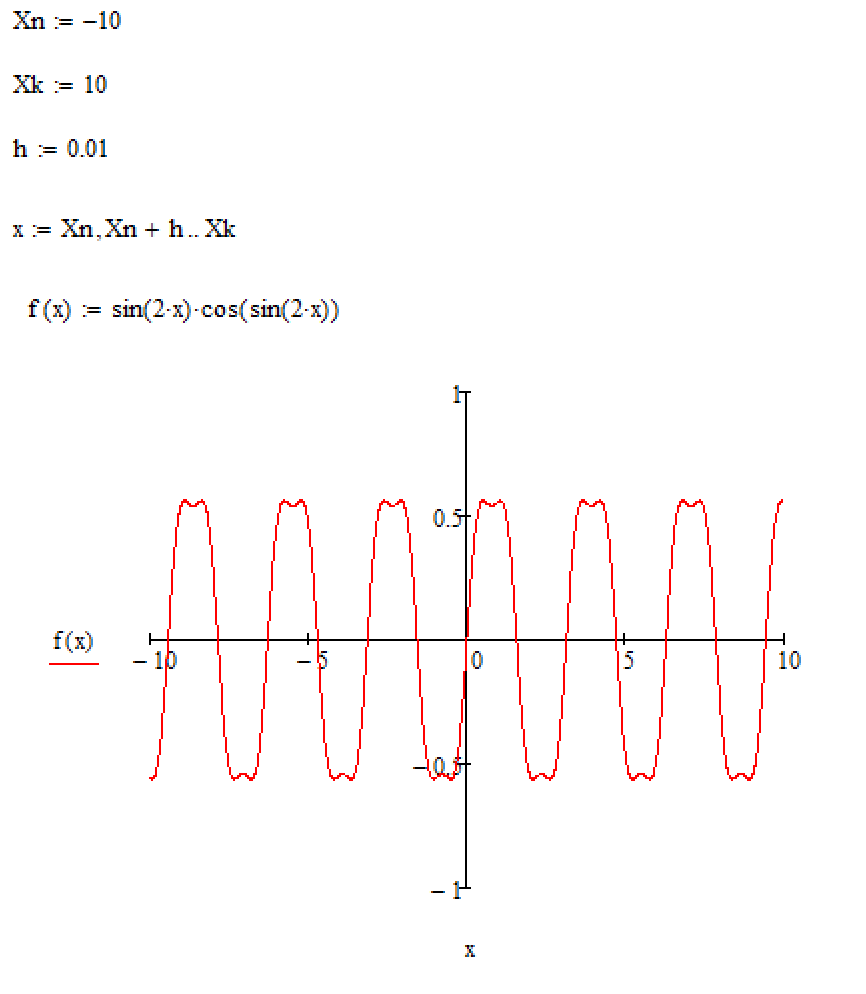


Рисунок. 15 – Проверка работы в *Mathcad*

На рисунке результат проверки вывода программы – график функции

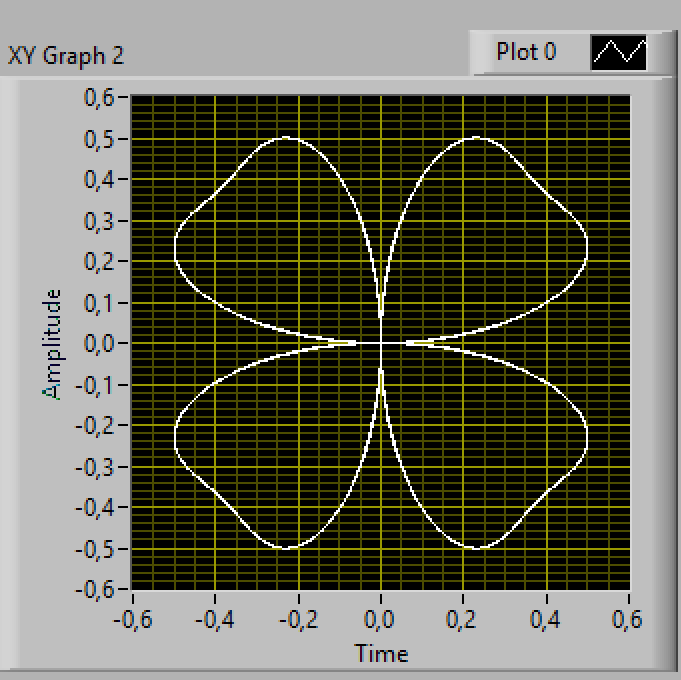


Рисунок. 16 – Результат работы программы в *LabView*

На рисунке результат вывода программы – график функции

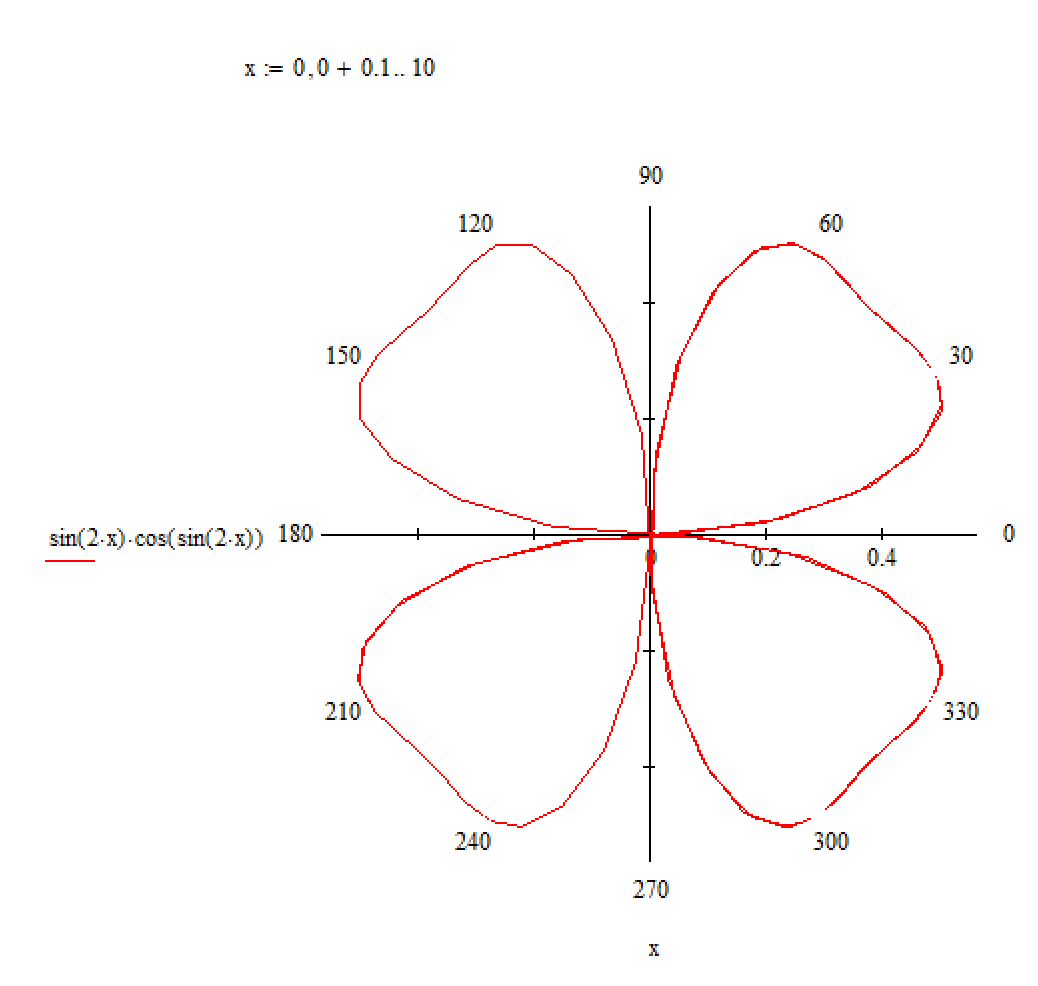


Рисунок. 17 – Проверка работы в *Mathcad*

На рисунке результат проверки вывода программы – график функции

* 1. Блок-схема:

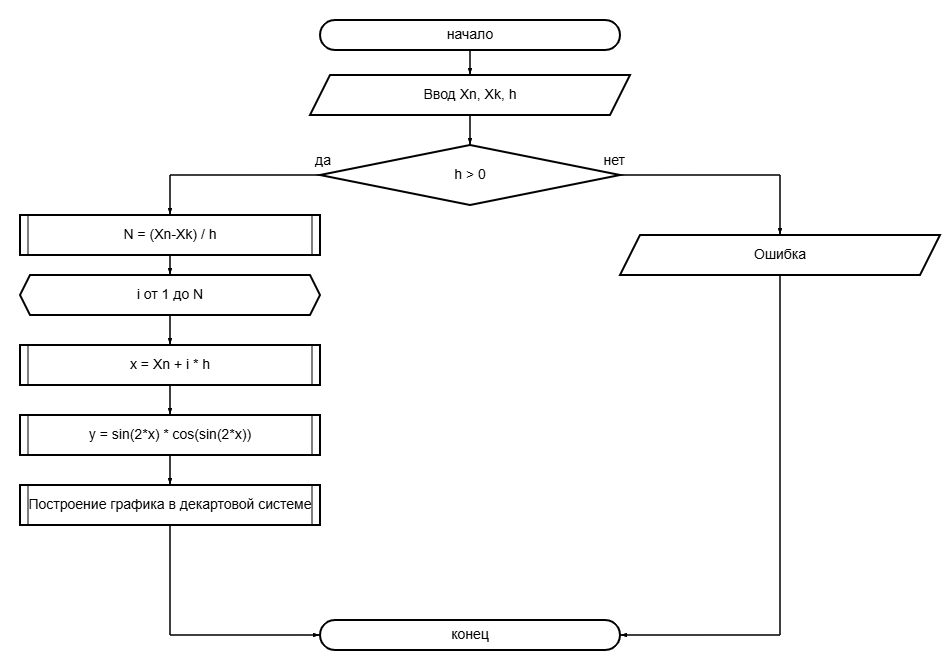


Рисунок. 18 – Блок-схема к заданию (декартова система)

На данном рисунке представлена блок-схема для построения графика в декартовой системе координат.

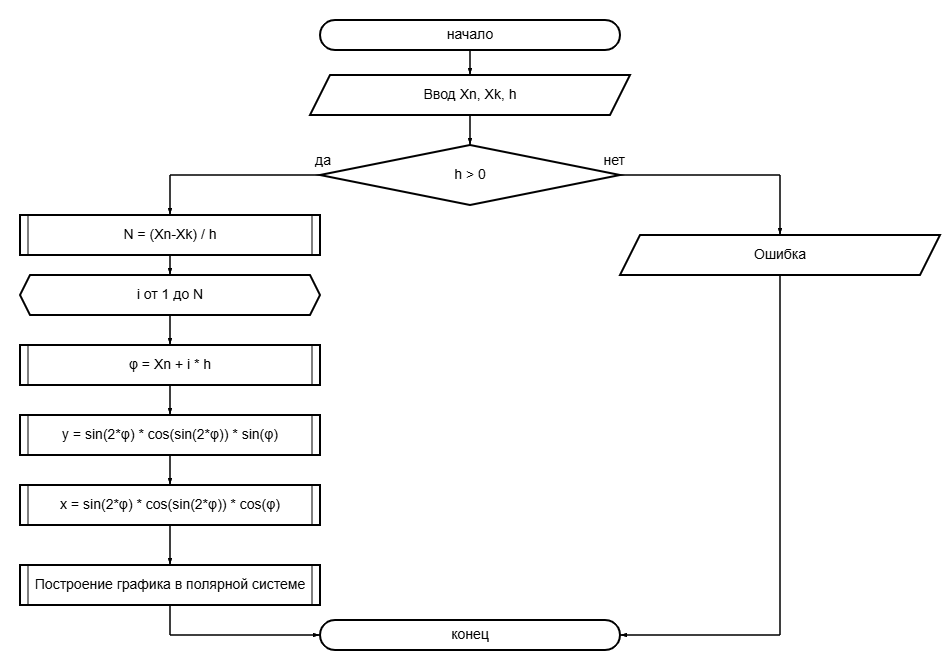


Рисунок. 19 – Блок-схема к заданию (полярная система)

На данном рисунке представлена блок-схема для построения графика в полярной системе координат.

* 1. Вывод:

Функционал программы *LabView* так же позволяет реализовать работу с графиками различных функций и корректно отображает полученный значения.

1. Задание № 3
   1. Цель работы:

Освоение навыков работы с текстовыми файлами как протоколами передачи информации между средами программирования и пакетами прикладных программ; приобретение знаний, касающихся настройки диалогов и событий; знакомство с механизмом обработки событий в среде с графическим языком программирования.

* 1. Условие задания:

В пакете прикладных программ *National Instruments LabView* создать виртуальные приборы для загрузки данных из файла и сохранения данных в файл.

* 1. Программа:

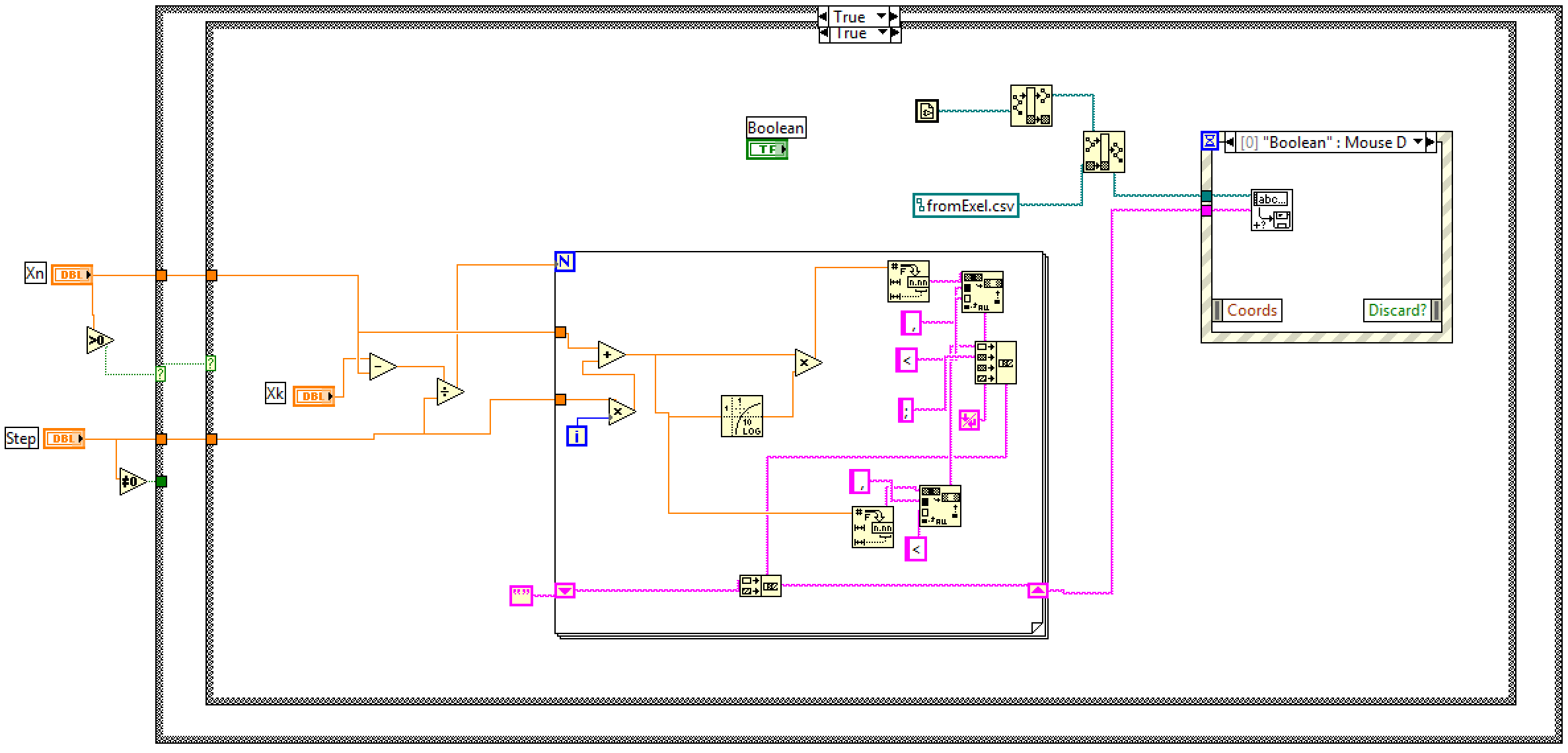


Рисунок. 20 – программа *second\_p.vi* записывающая данные в файл

На данном рисунке полностью представлена программа, осуществляющая запись значений x и y в файл под названием *fromExel.csv.*

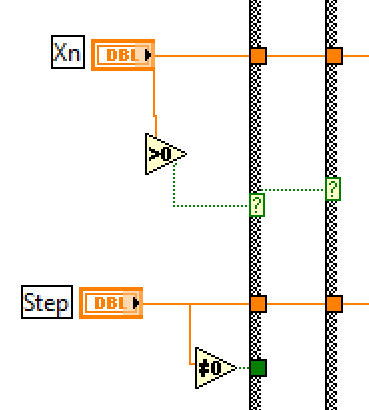


Рисунок. 21 – программа *second\_p.vi* (проверка шага и начального значения)

На данном рисунке представлена проверка шага и начального значения. Если начальное значение меньше нуля – выдается ошибка из-за условия логарифма. Шаг не должен быть равен нулю.

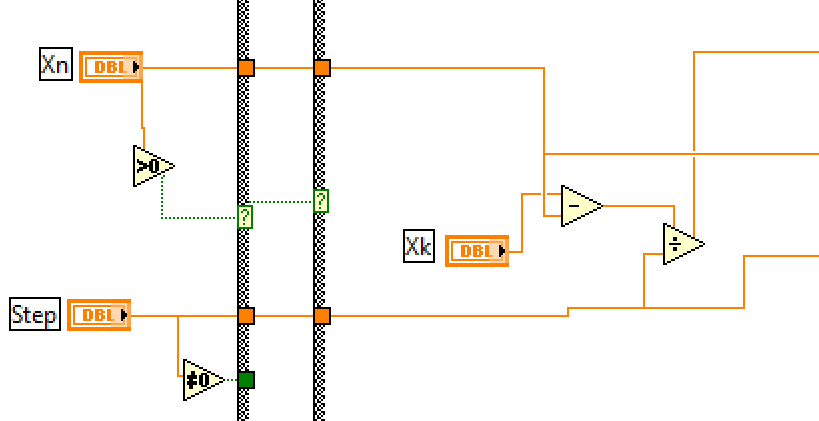


Рисунок. 22 – программа *second\_p.vi* (Получение количества значений)

На данном рисунке представлен расчет количества значений (*N*). *N* = *(Xn – Xk) / h.*

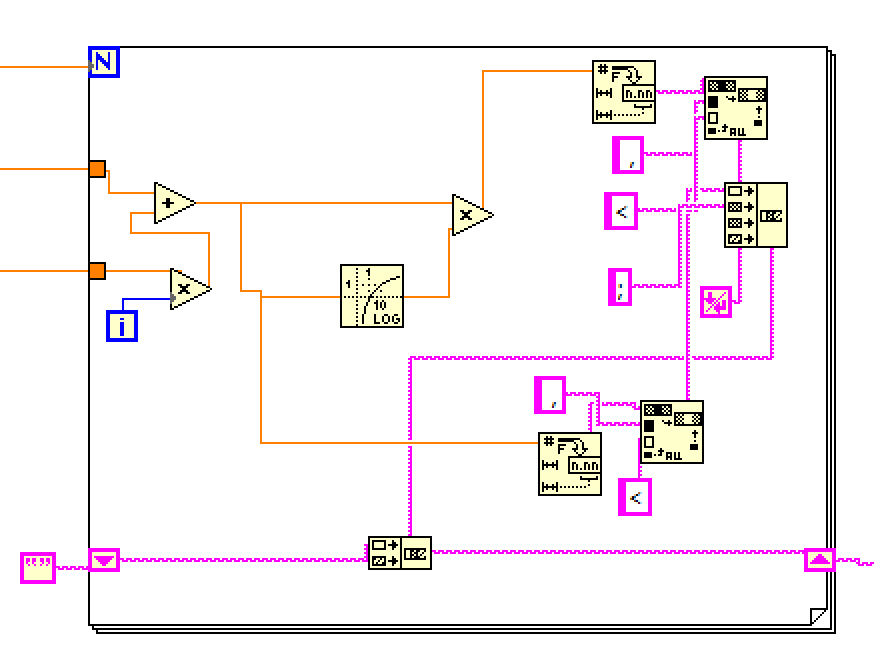


Рисунок. 23 – программа *second\_p.vi* (Получение значений функции в виде строки с установленными разделителями)

На данном рисунке представлена программа, принимающая шаг, количество значений, и начальное значение. *x = i \* step + Xn.* Далее происходит расчет *y*, . Далее эти параметры преобразуются в строку так, что разделительные знаки заменяются на “<” , между значениями ставится “;”, для того чтобы Excel понимал когда перенести значение в соседнюю ячейку и знак переноса строки вниз.

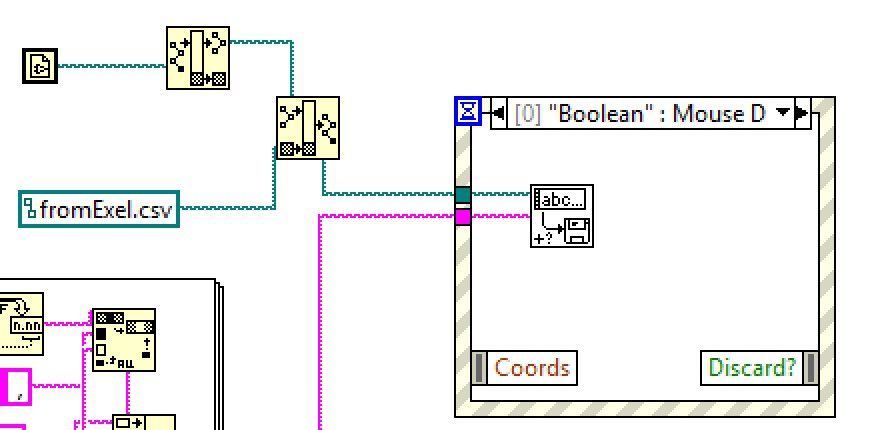


Рисунок. 23 – программа *second\_p.vi* (Запись значений в файл)

На данном рисунке происходит запись значений в файл. Сам файл открывается через путь.

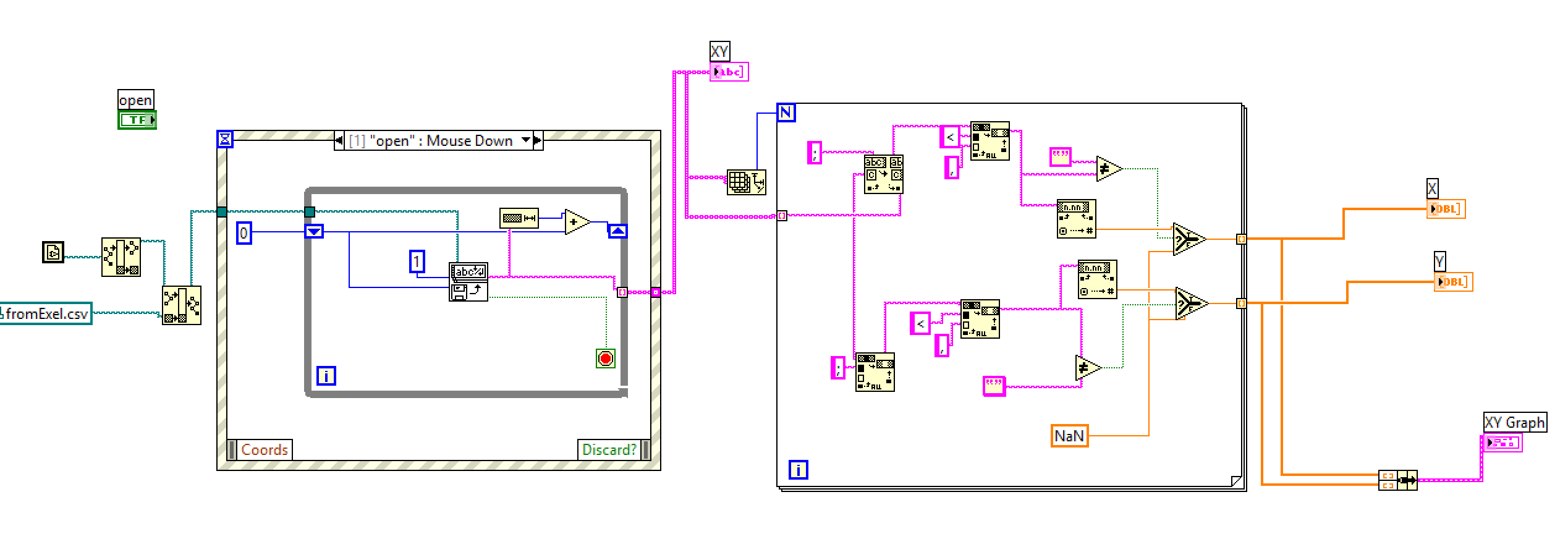


Рисунок. 24 – программа *first\_p.vi* (полностью) строящая график из файла

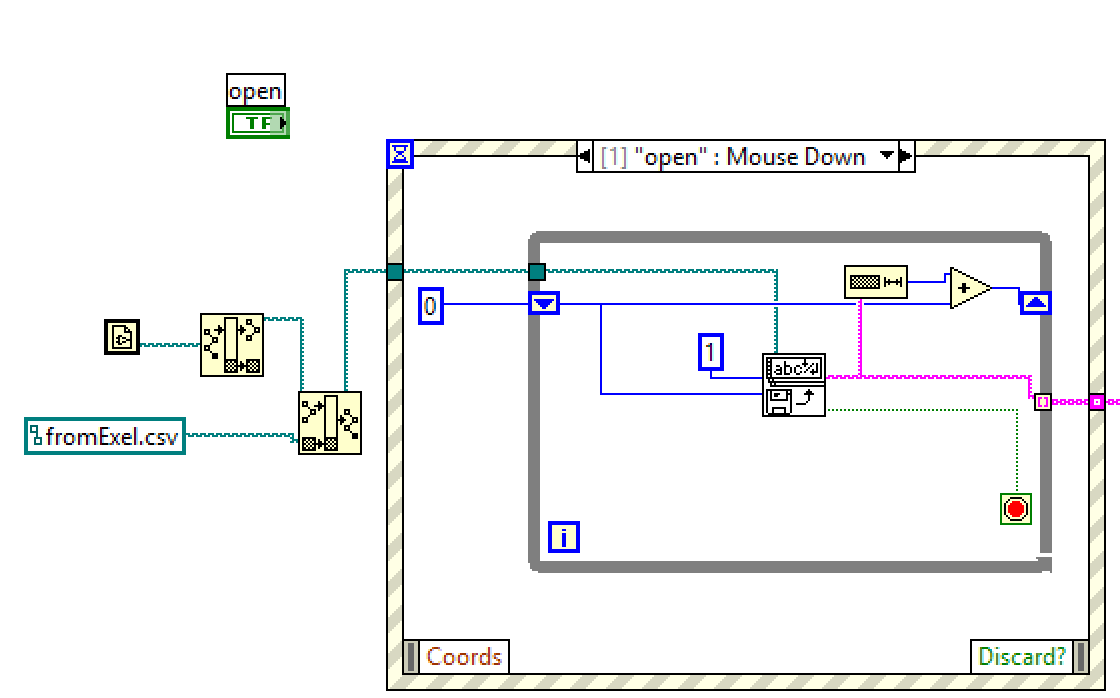


Рисунок. 25 – программа *first\_p.vi*(открытие файла и преобразование в массив)

На данном рисунке показана программа, открывающая файл *fromExel.csv* и преобразующая значения в массив.

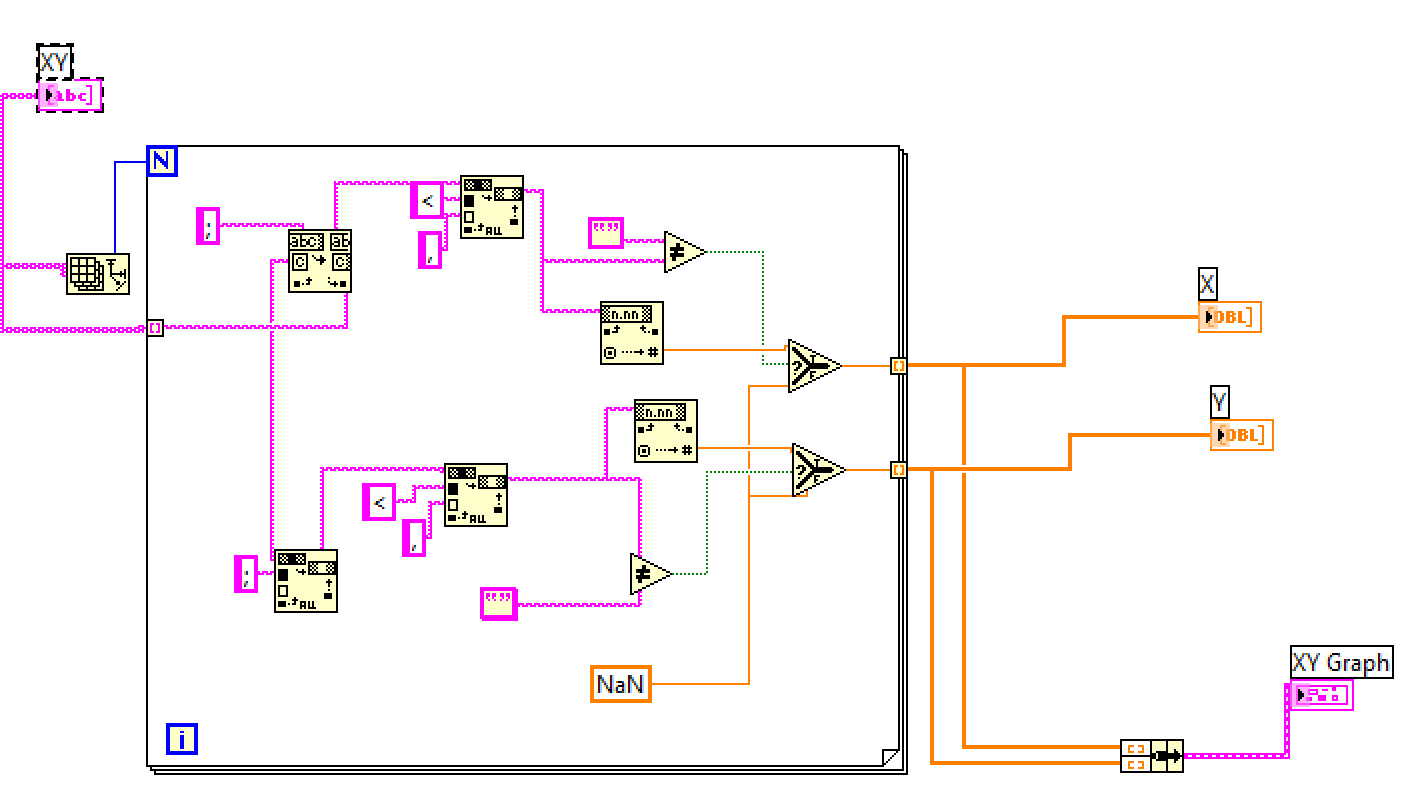


Рисунок. 26 – программа *first\_p.vi*(построение графика из массива)

На данном рисунке из массива *XY* получаем размер массива(*N*). Далее происходит разбитие строки массива на 2 строки чрез разделитель “;”, что было перед ним – значение координаты *x*, а после – *y,* далее происходит обратная замена разделителей с “>” на “,”, для того чтобы преобразовать значения из строк к числам, на основе которых и строится график.

* 1. Проверка результатов:

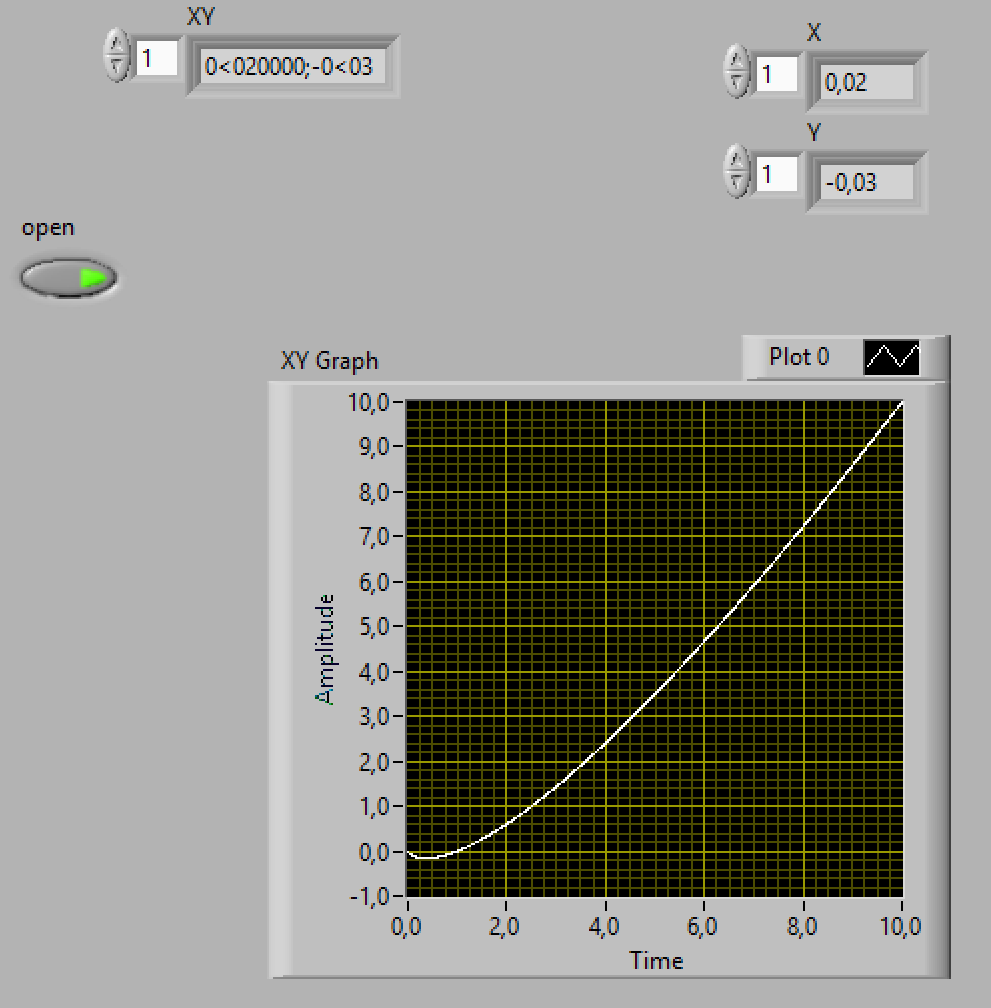


Рисунок. 27 – Результат вывода в *LabView*

На данном рисунке результат работы программы в *LabView,* где *XY –* строковый массивзначений. X и Y – уже преобразованные координаты из строки. *XY Graph* – график по точкам.

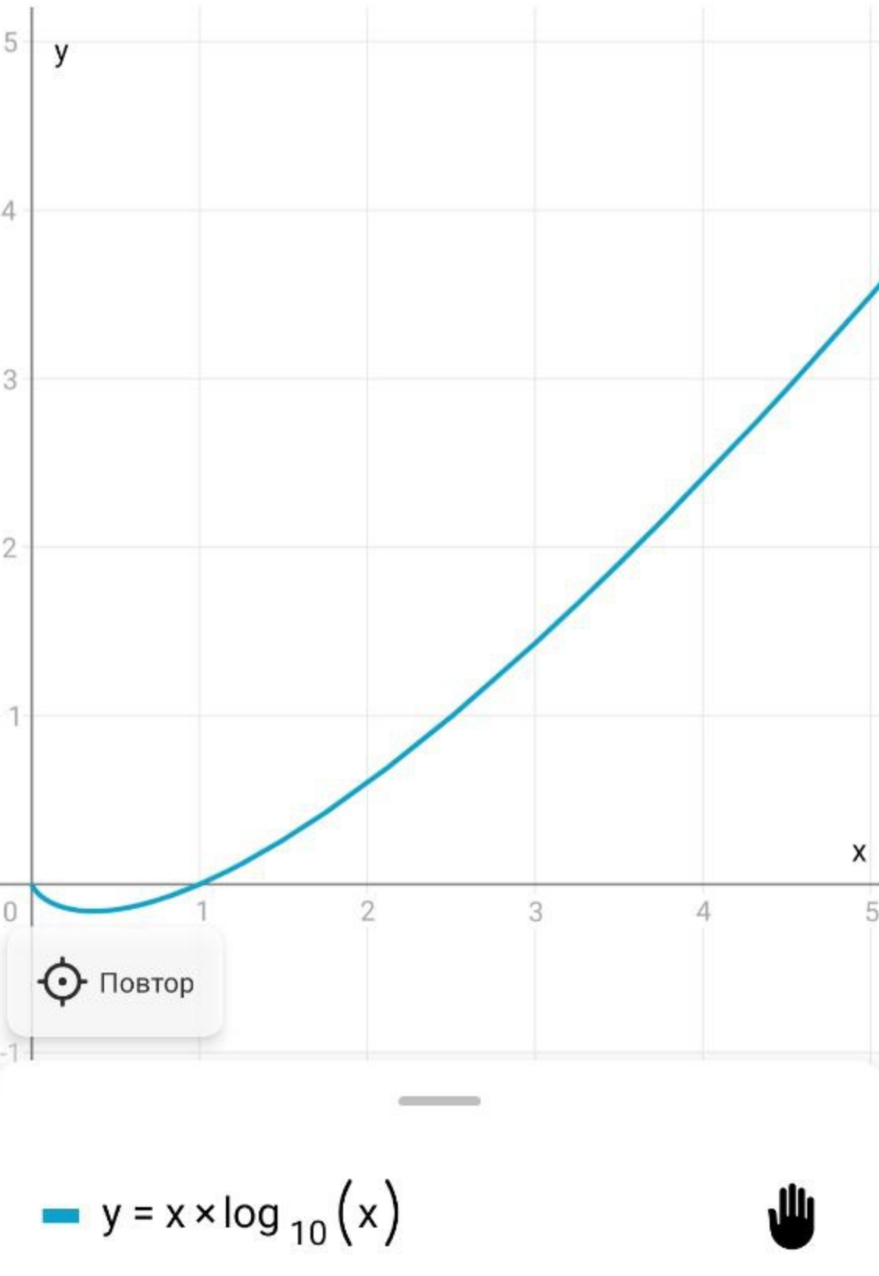


Рисунок. 28 – Проверка в *Photomath*

На данном рисунке результат проверки функции в *Photomath.*

* 1. Блок-схема:

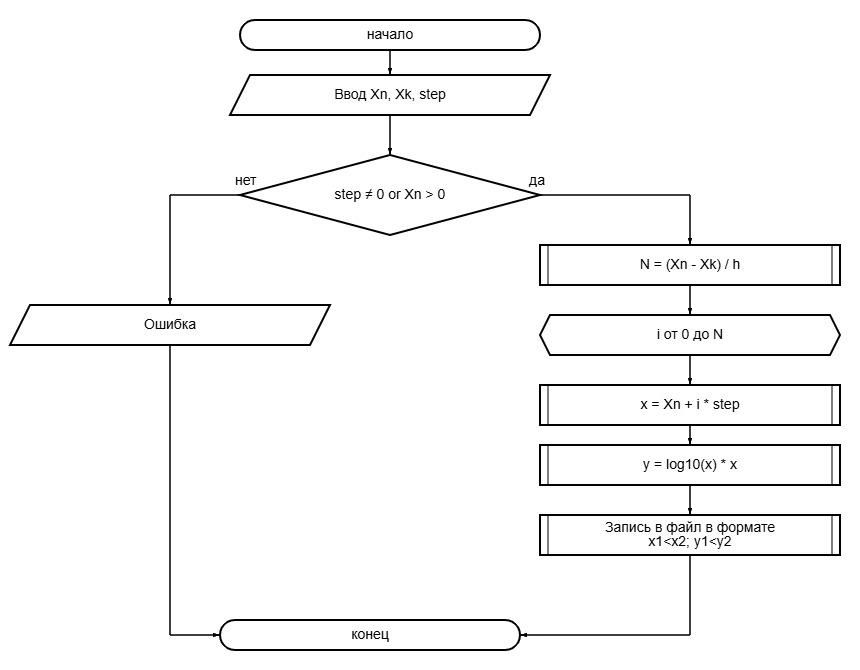


Рисунок. 29 – Блок-схема *second\_p.vi*

На данном рисунке представлена блок-схема *second\_p.vi.*

Примечание: *x1, y1* – целые части x и y соответственно. *x2, y2* – дробные части x и y соответственно.

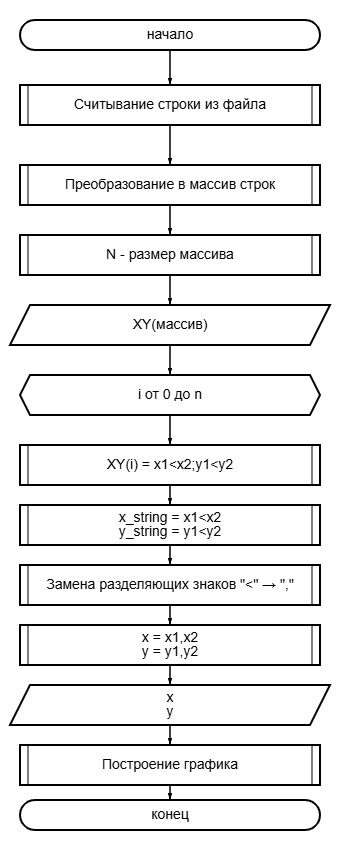


Рисунок. 30 – Блок-схема *first\_p.vi*

На данном рисунке представлена блок-схема *first\_p.vi,* берущая значения из файла и строящая по ним график*.*

Примечание: *x1, y1* – целые части x и y соответственно. *x2, y2* – дробные части x и y соответственно.

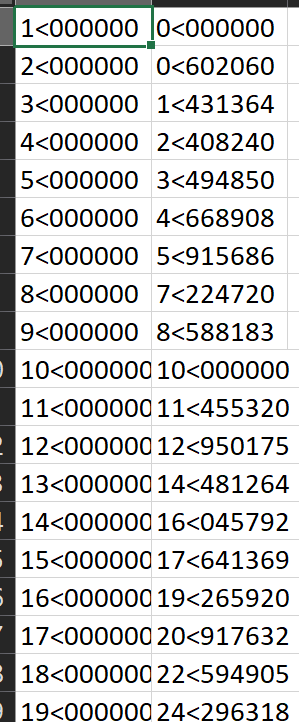


Рисунок. 31 – Формат данных в *Excel*

На данном рисунке представлены координаты точек в файле.

* 1. Вывод:

Функционал программы *LabView* позволяет работать с другими файлами, записывать значения в *Excel*, задавать произвольные разделители. Так же можно считывать данные из различных файлов и использовать их для решения задач.

1. Задание № 4
   1. Цель работы:

Освоение навыков работы с локальными и глобальными переменными, а также структурой последовательного вычислительного процесса внутри системы, адаптированной под потоковые вычисления.

* 1. Условие задания:

В пакете прикладных программ *National Instruments LabView* создать виртуальный прибор, моделирующий поведение типовых сигнализаторов и индикаторов (заданных по вариантам).

* 1. Программа:

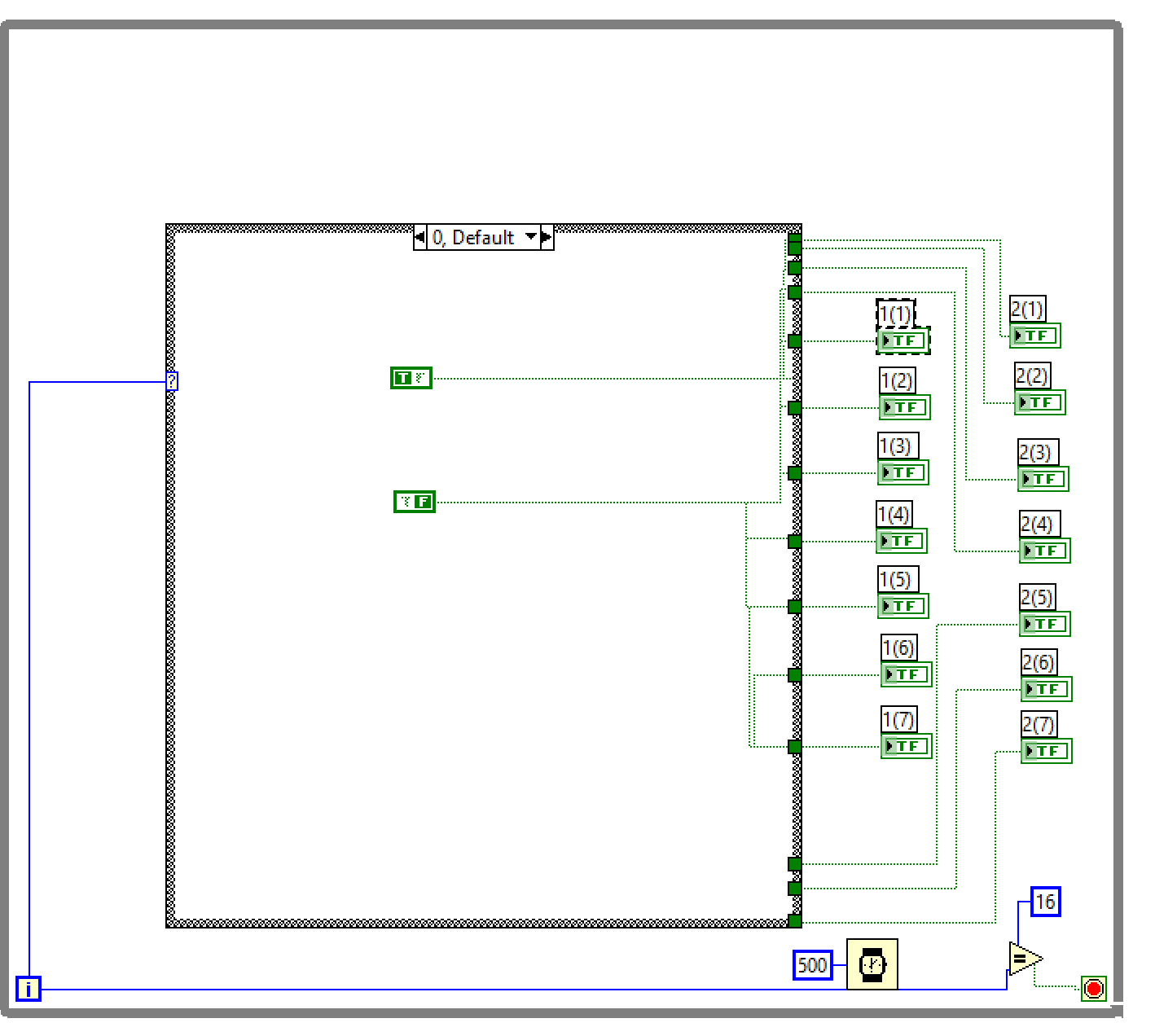


Рисунок. 32 – Программа *4.vi*(0 состояние)

На данном рисунке представлено 0 состояние из 16.

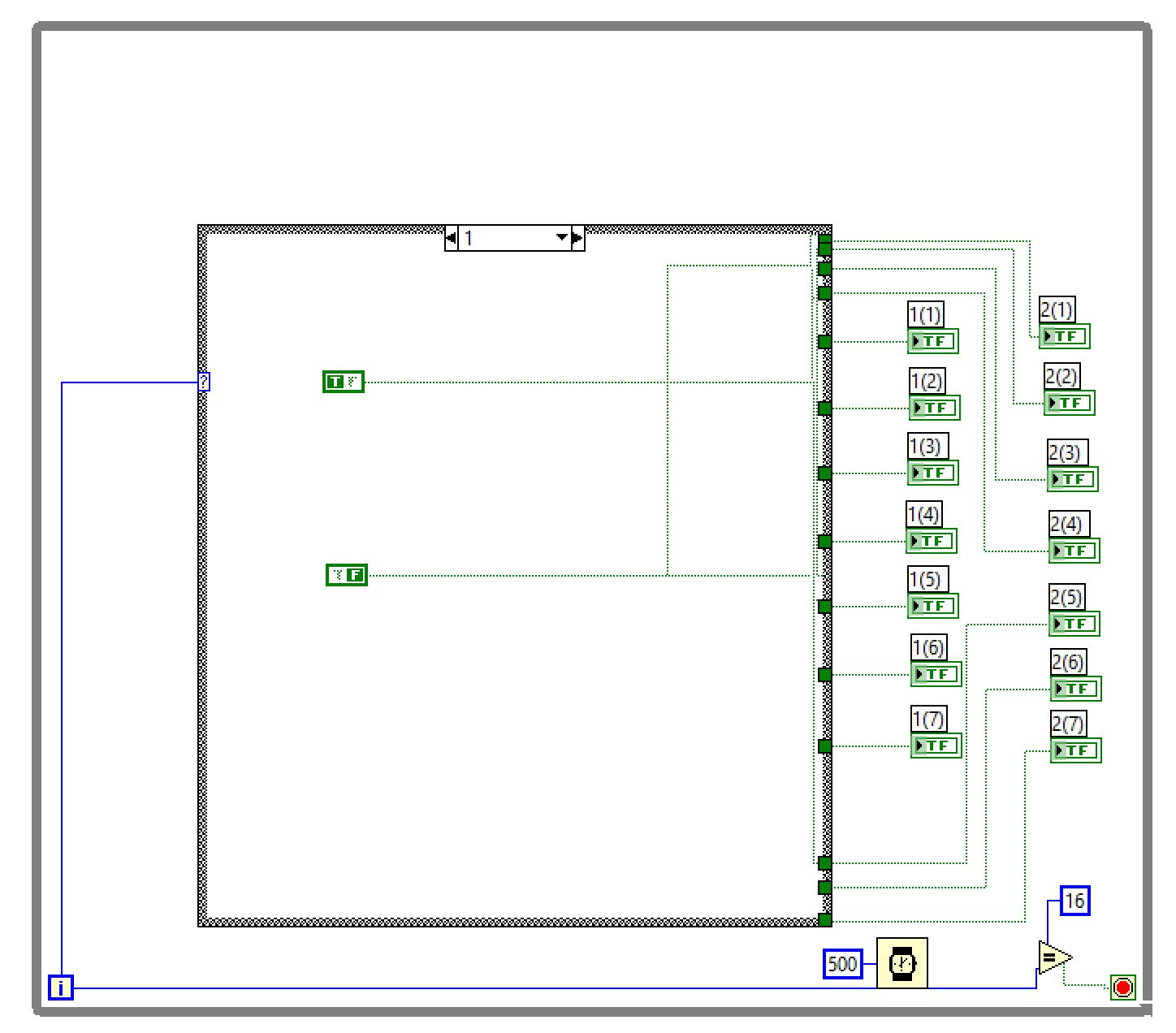


Рисунок. 33 – Программа *4.vi*(1 состояние)

На данном рисунке представлено 1 состояние из 16.

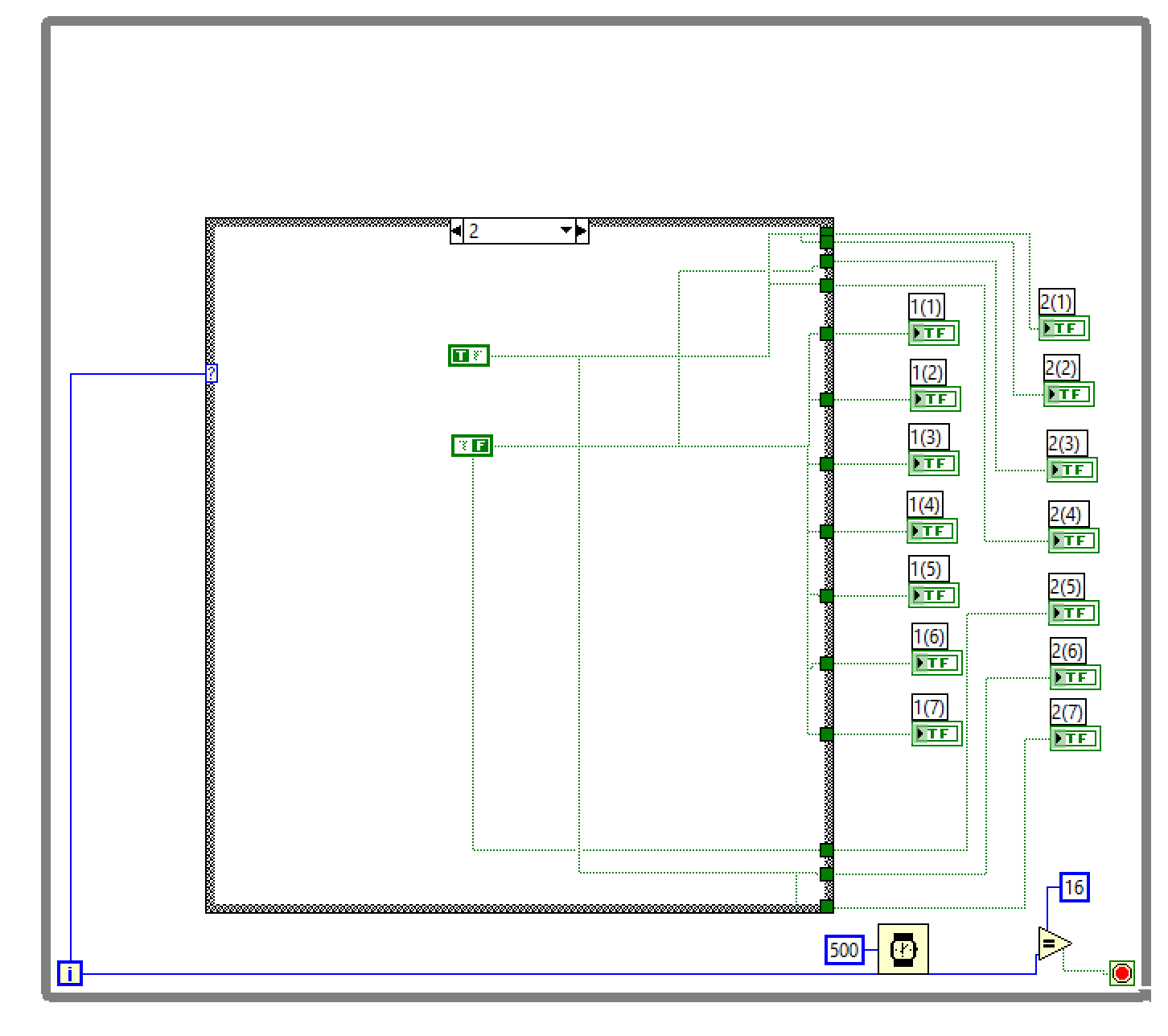


Рисунок. 34 – Программа *4.vi*(2 состояние)

На данном рисунке представлено 2 состояние из 16.

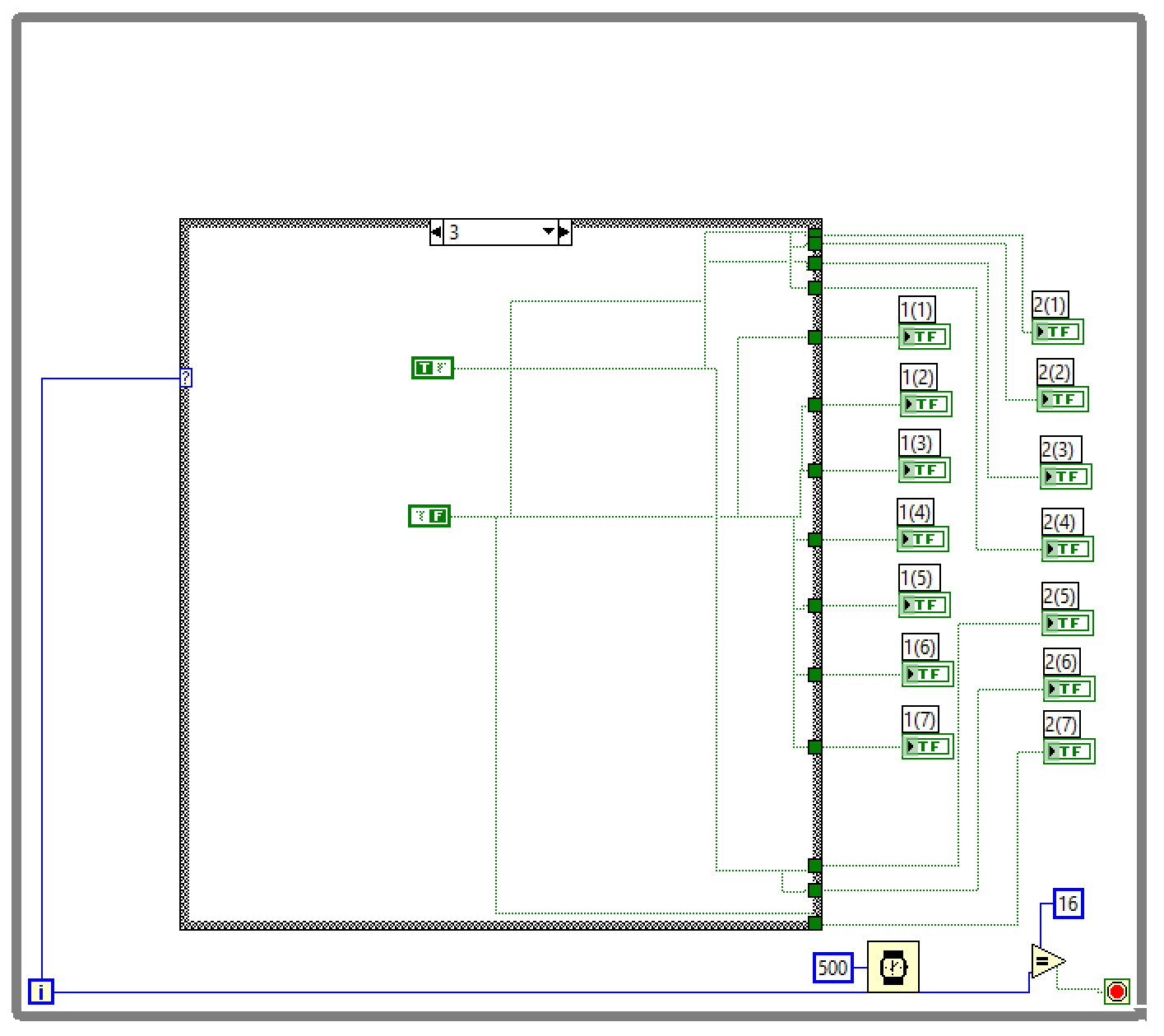


Рисунок. 35 – Программа *4.vi*(3 состояние)

На данном рисунке представлено 3 состояние из 16.

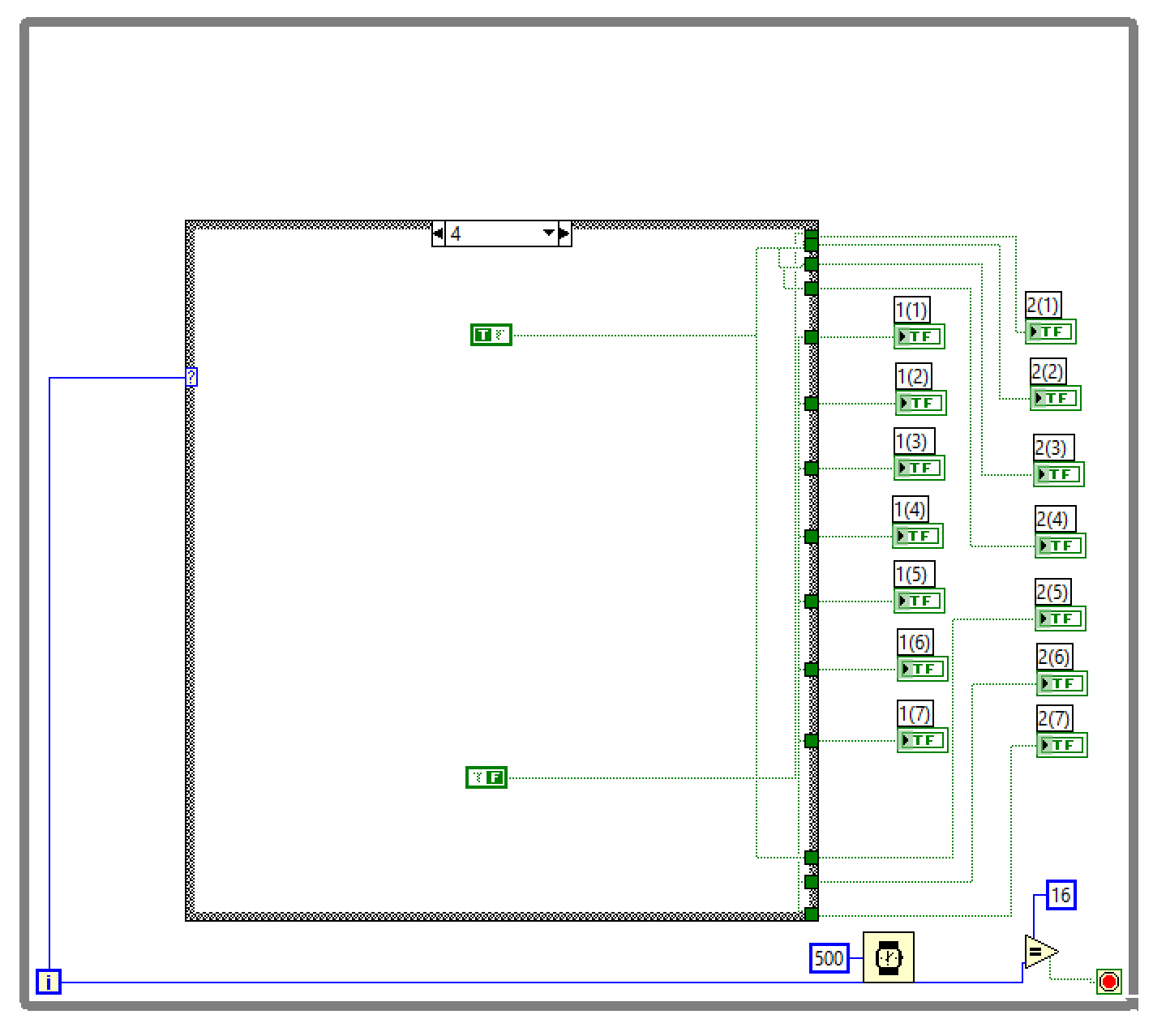


Рисунок. 36 – Программа *4.vi*(4 состояние)

На данном рисунке представлено 4 состояние из 16.

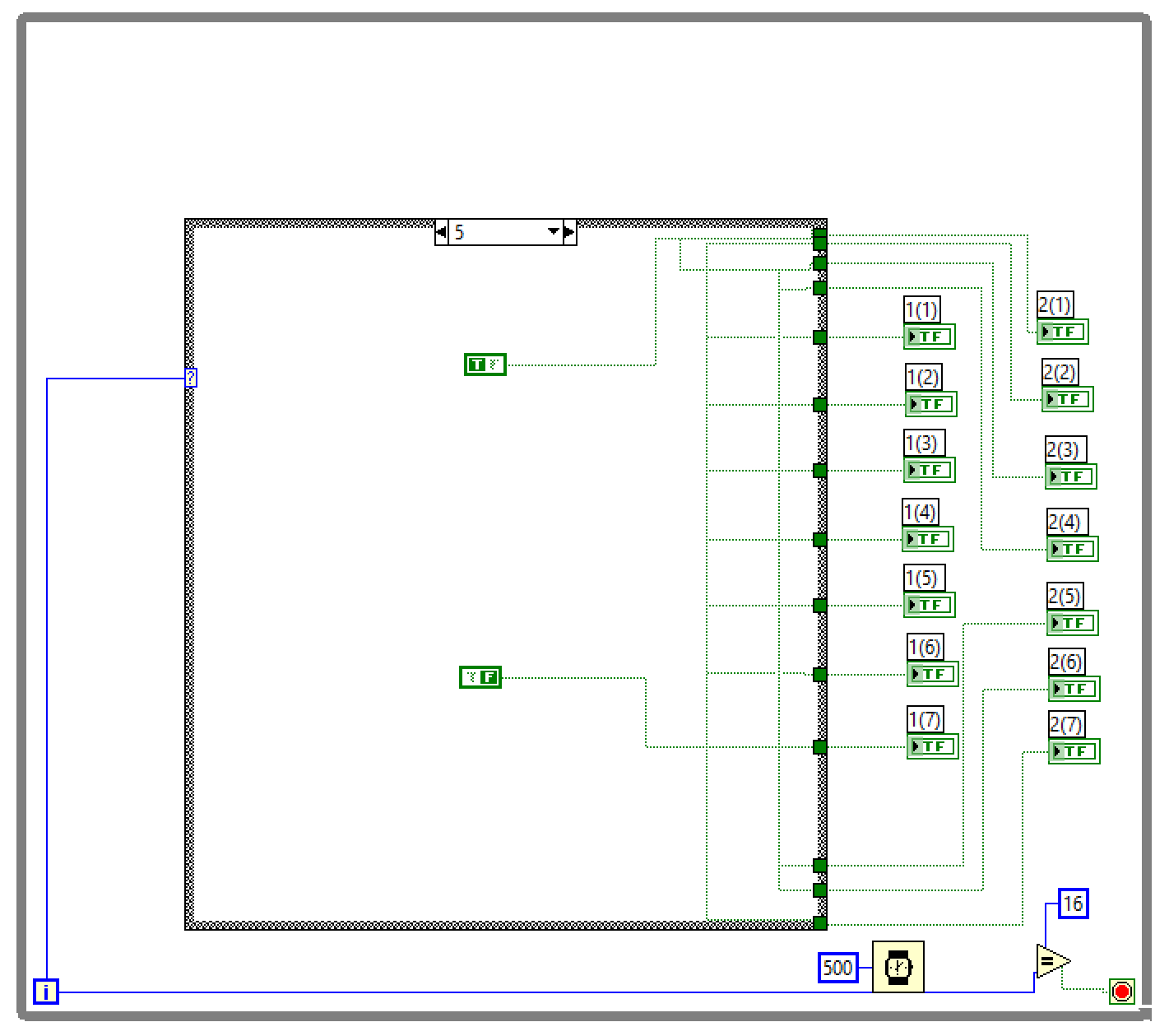
****

Рисунок. 37 – Программа *4.vi*(5 состояние)

На данном рисунке представлено 5 состояние из 16.

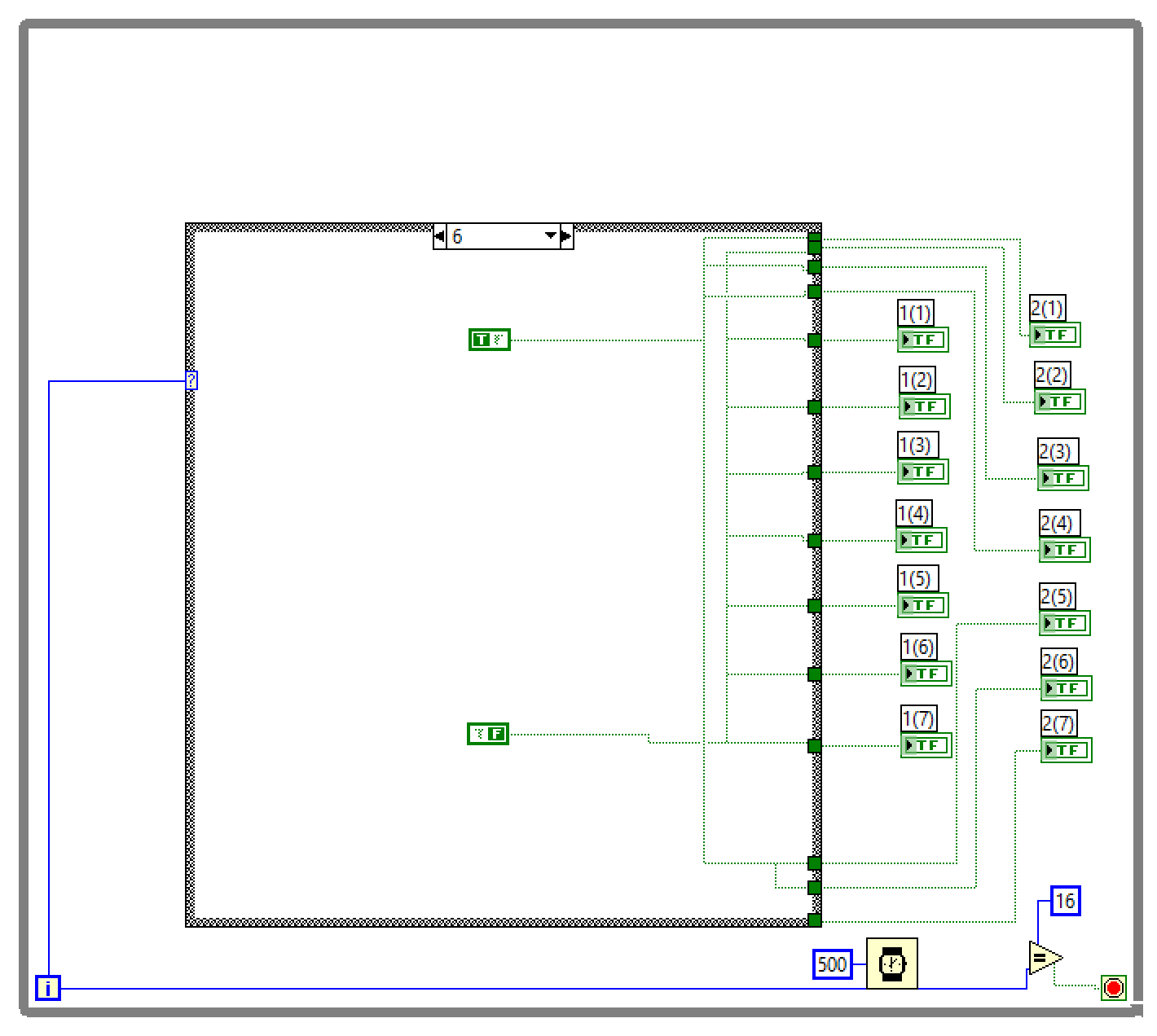


Рисунок. 38 – Программа *4.vi*(6 состояние)

На данном рисунке представлено 6 состояние из 16.

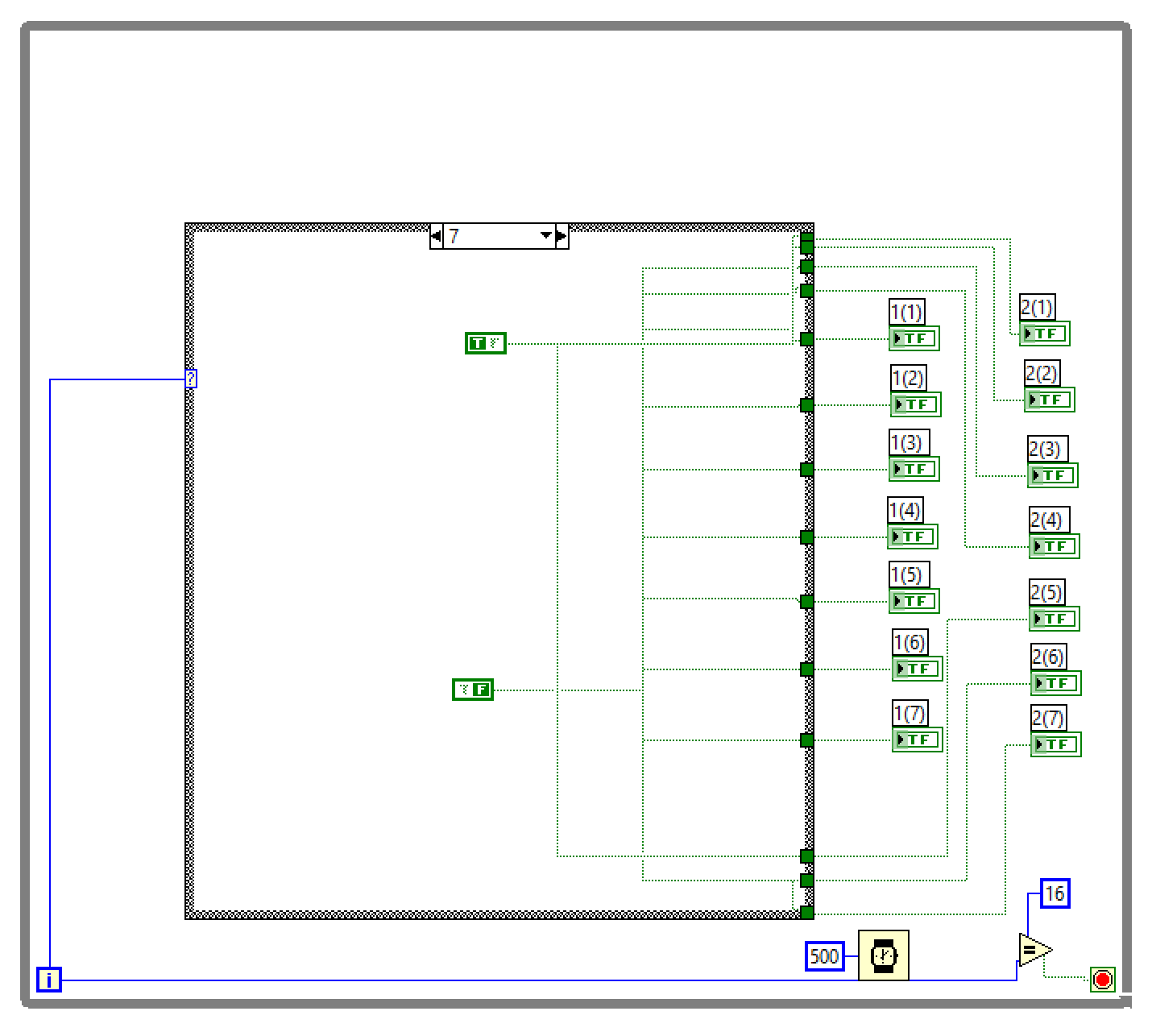


Рисунок. 39 – Программа *4.vi*(7 состояние)

На данном рисунке представлено 7 состояние из 16.

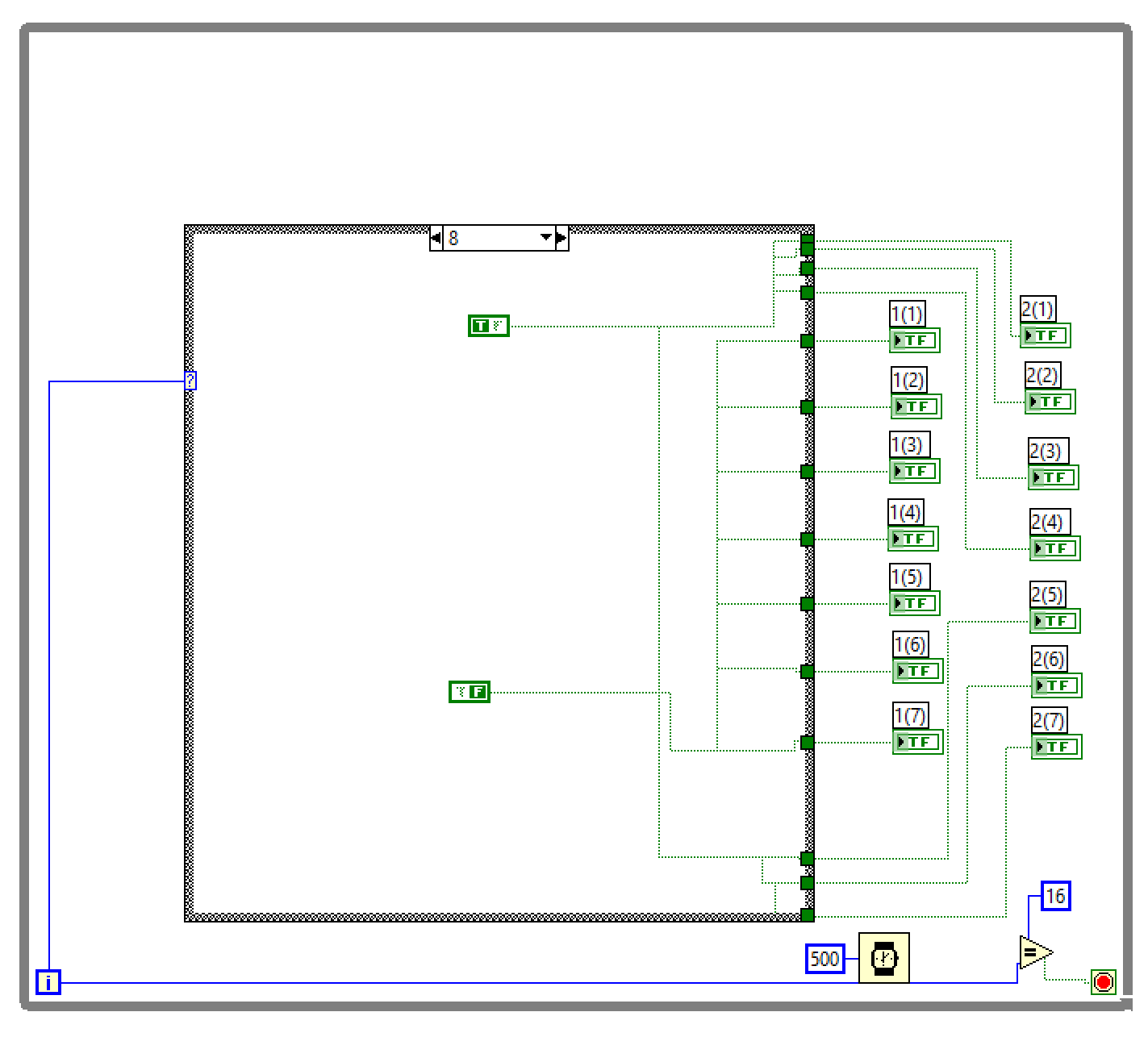


Рисунок. 40 – Программа *4.vi*(8 состояние)

На данном рисунке представлено 8 состояние из 16.

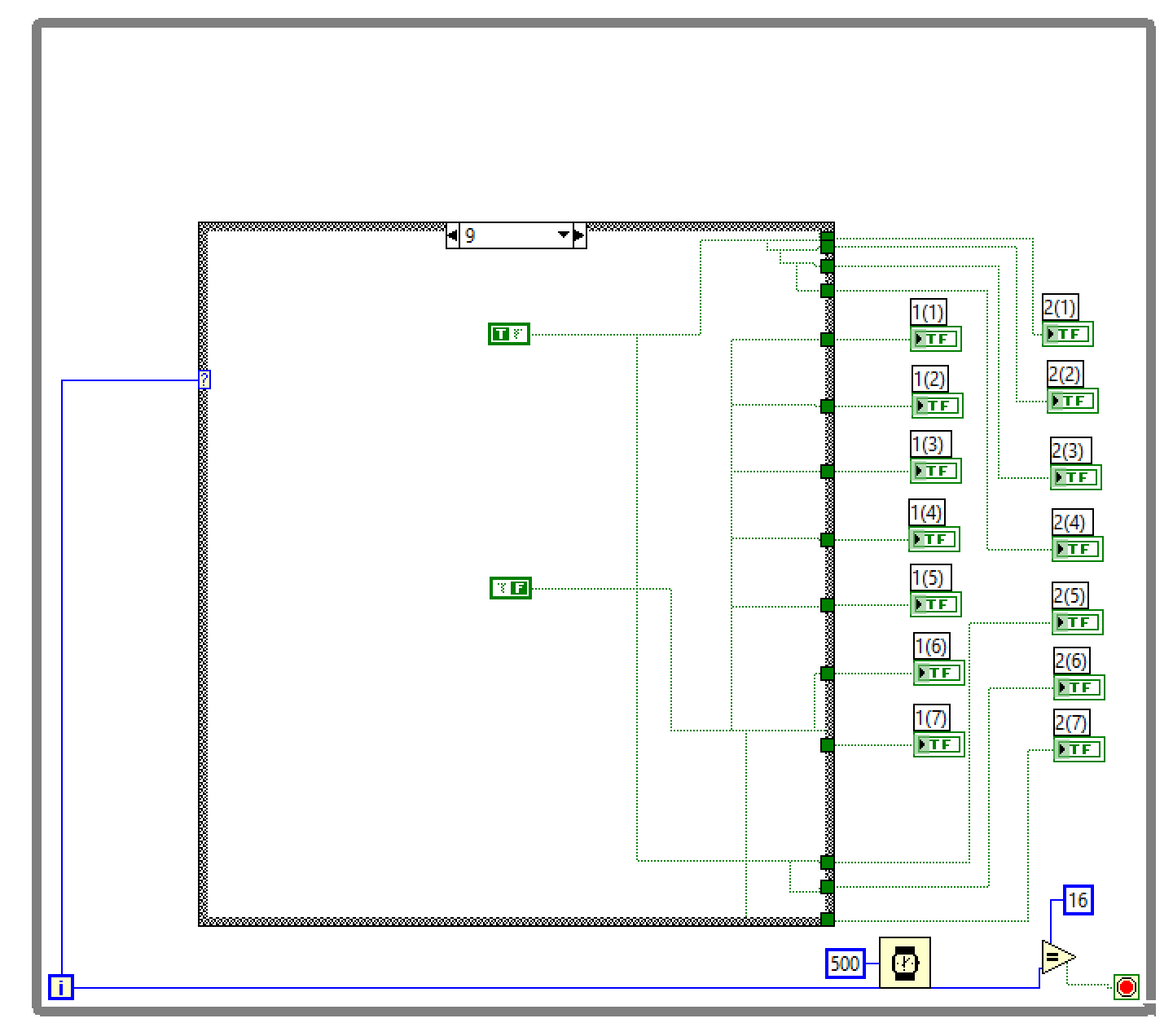


Рисунок. 41 – Программа *4.vi*(9 состояние)

На данном рисунке представлено 9 состояние из 16.

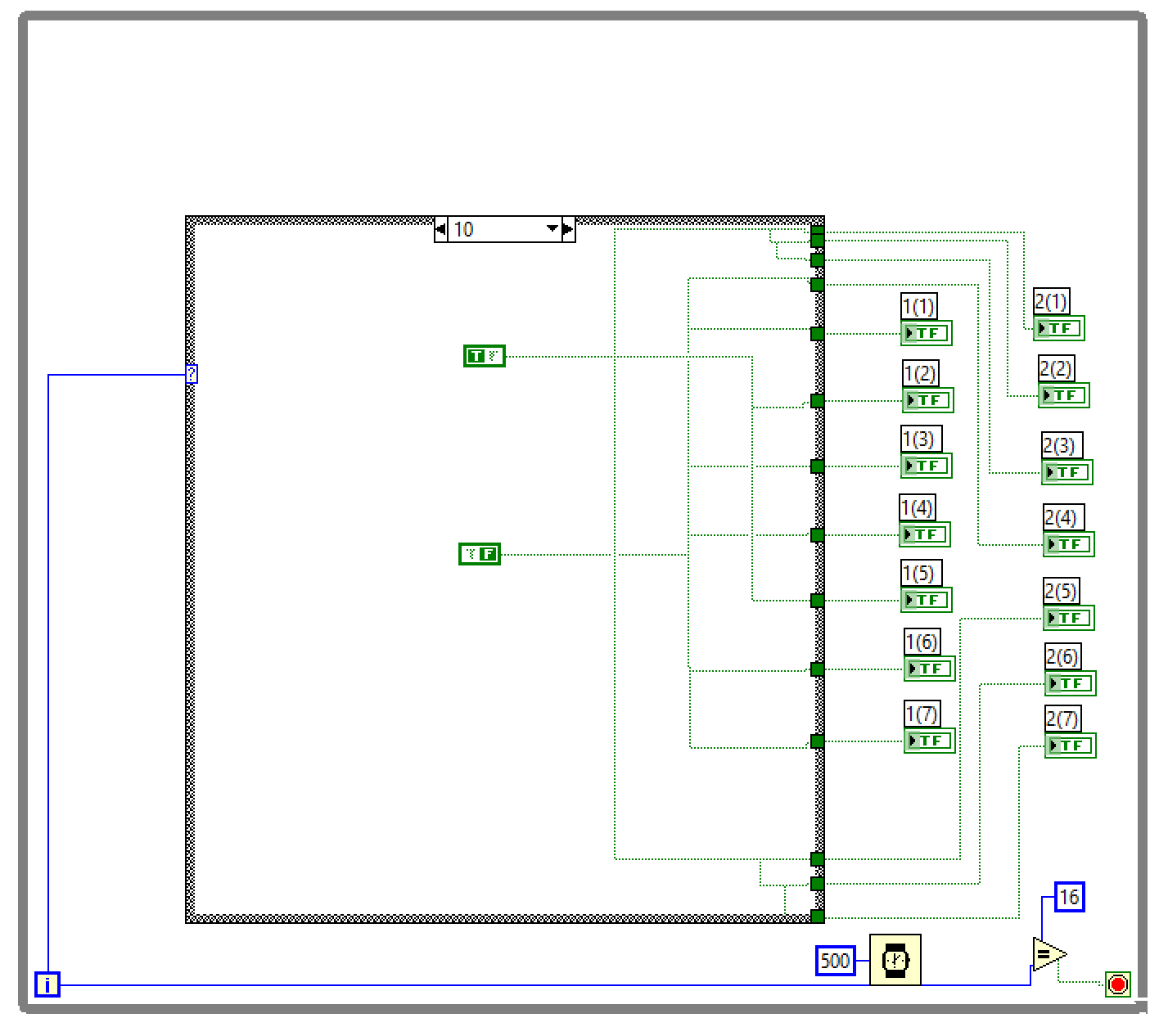


Рисунок. 42 – Программа *4.vi*(10 состояние)

На данном рисунке представлено 10 состояние из 16.

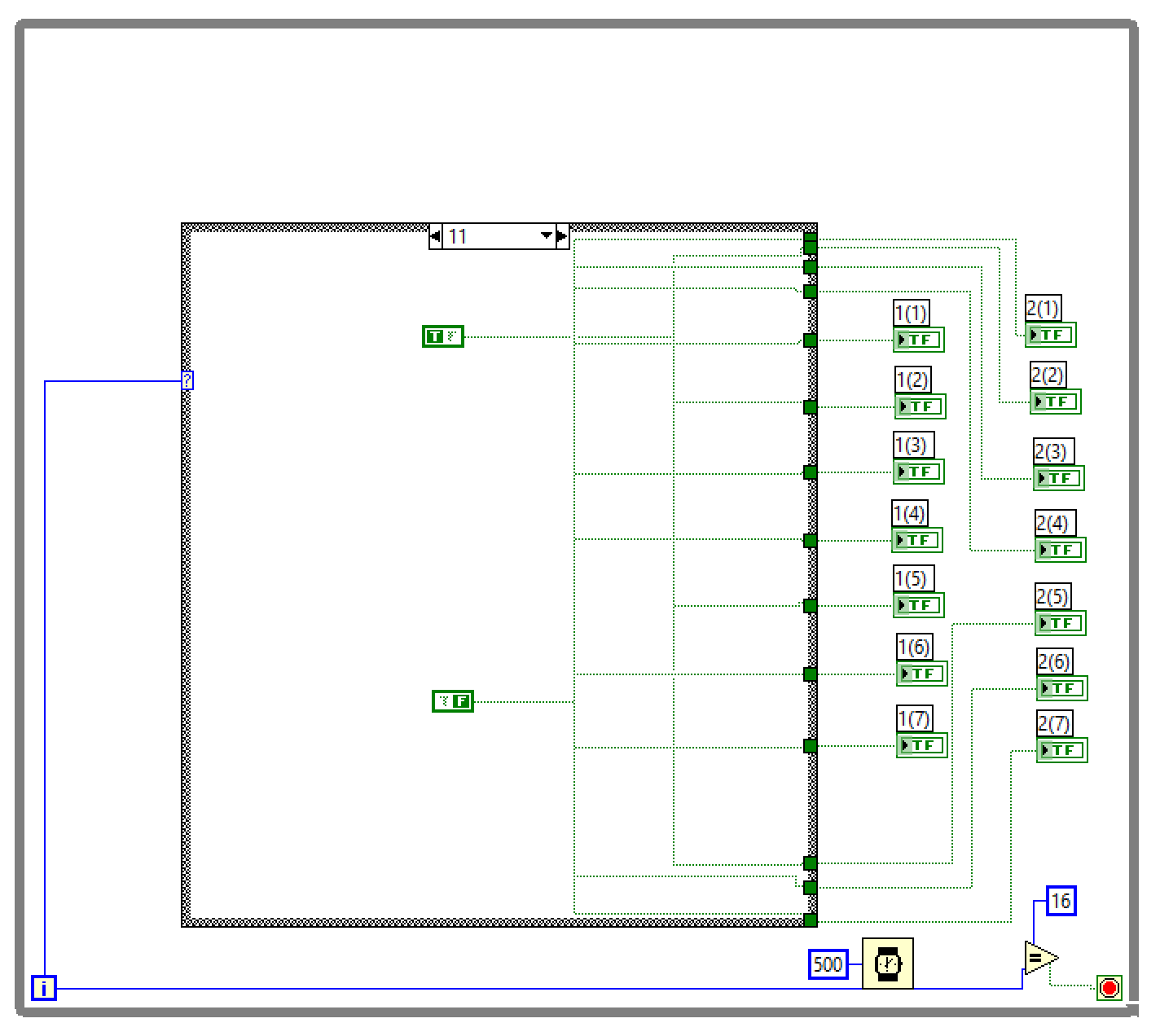


Рисунок. 43 – Программа *4.vi*(11 состояние)

На данном рисунке представлено 11 состояние из 16.

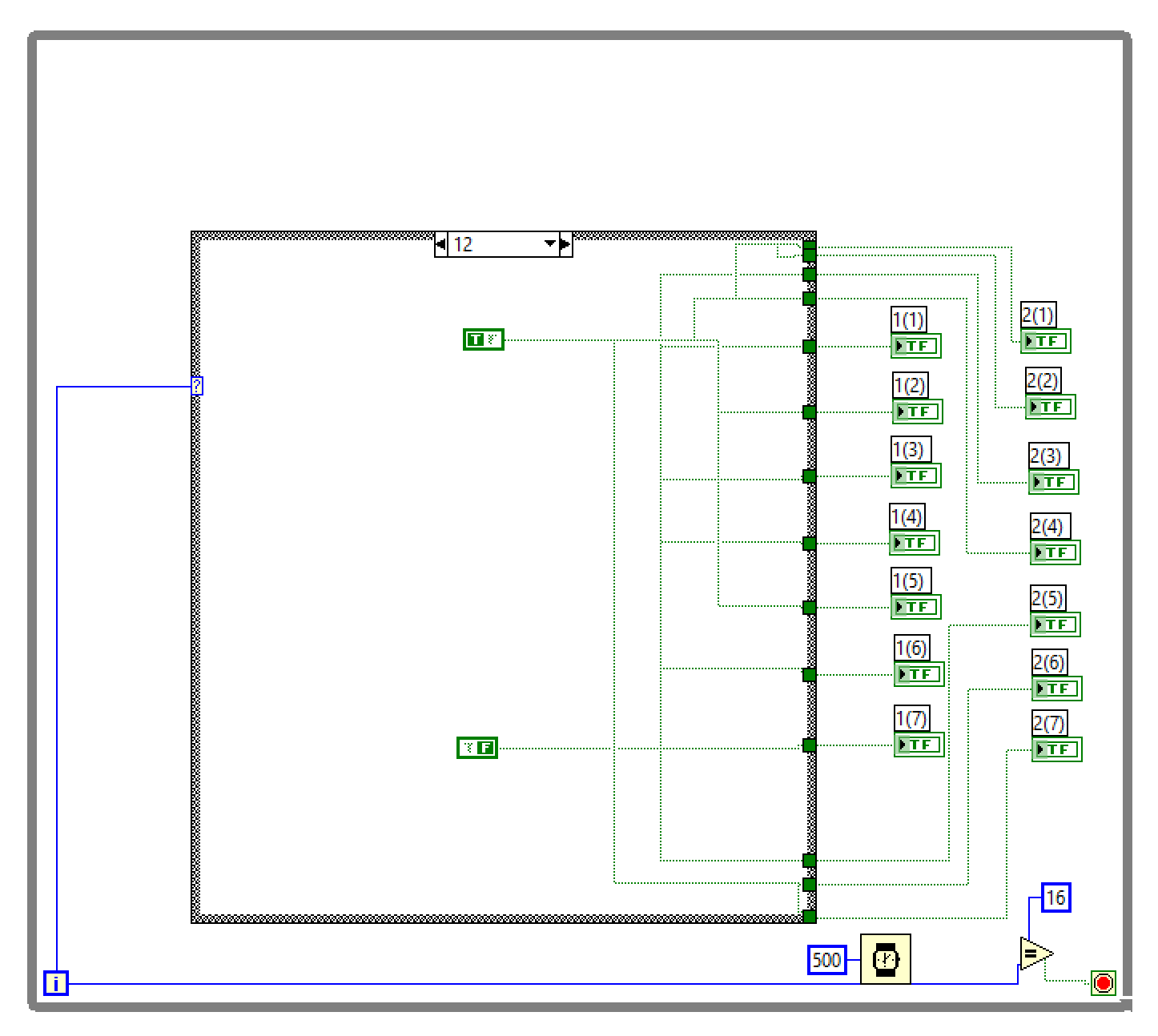


Рисунок. 44 – Программа *4.vi*(12 состояние)

На данном рисунке представлено 12 состояние из 16.

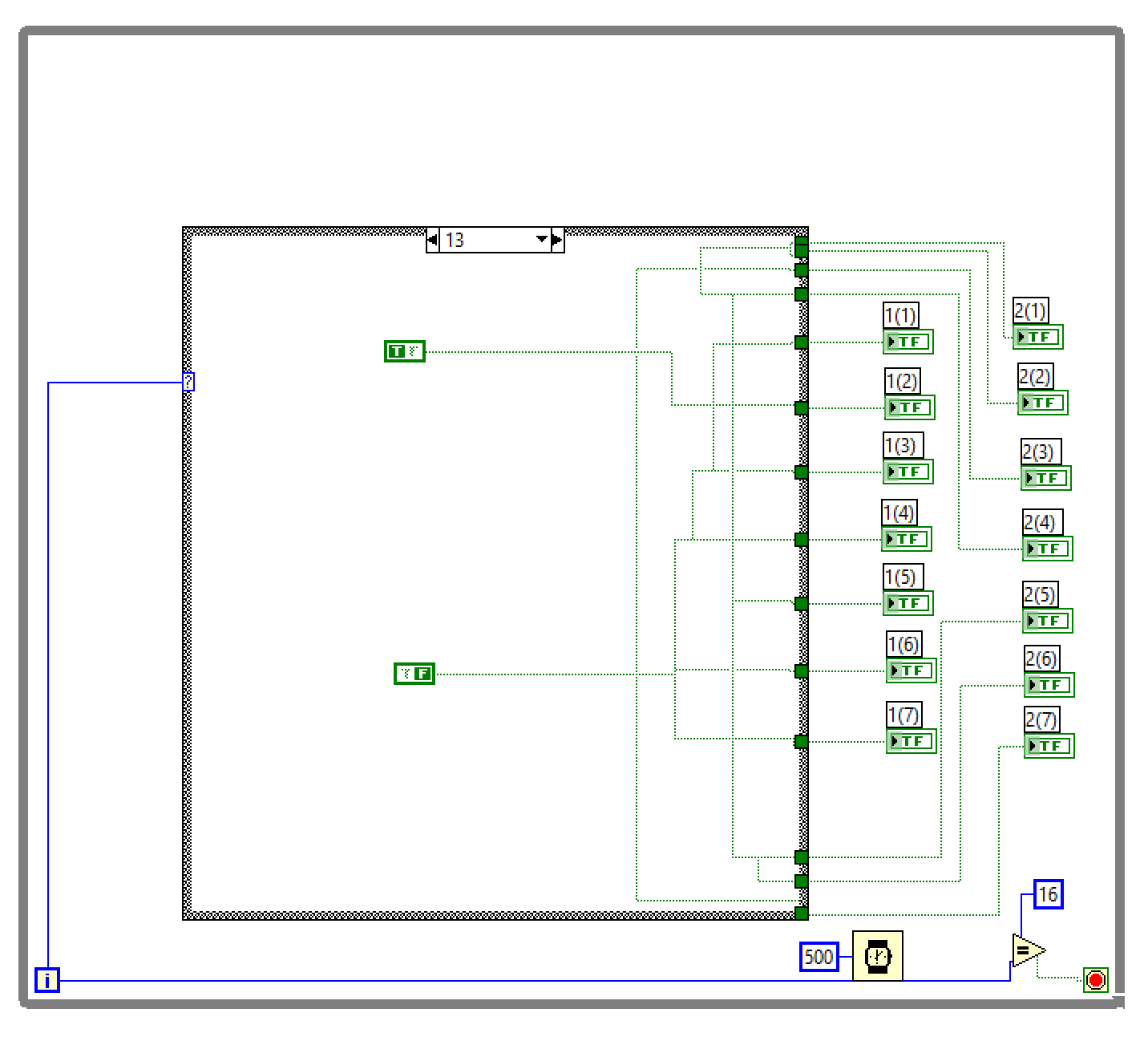


Рисунок. 45 – Программа *4.vi*(13 состояние)

На данном рисунке представлено 13 состояние из 16.

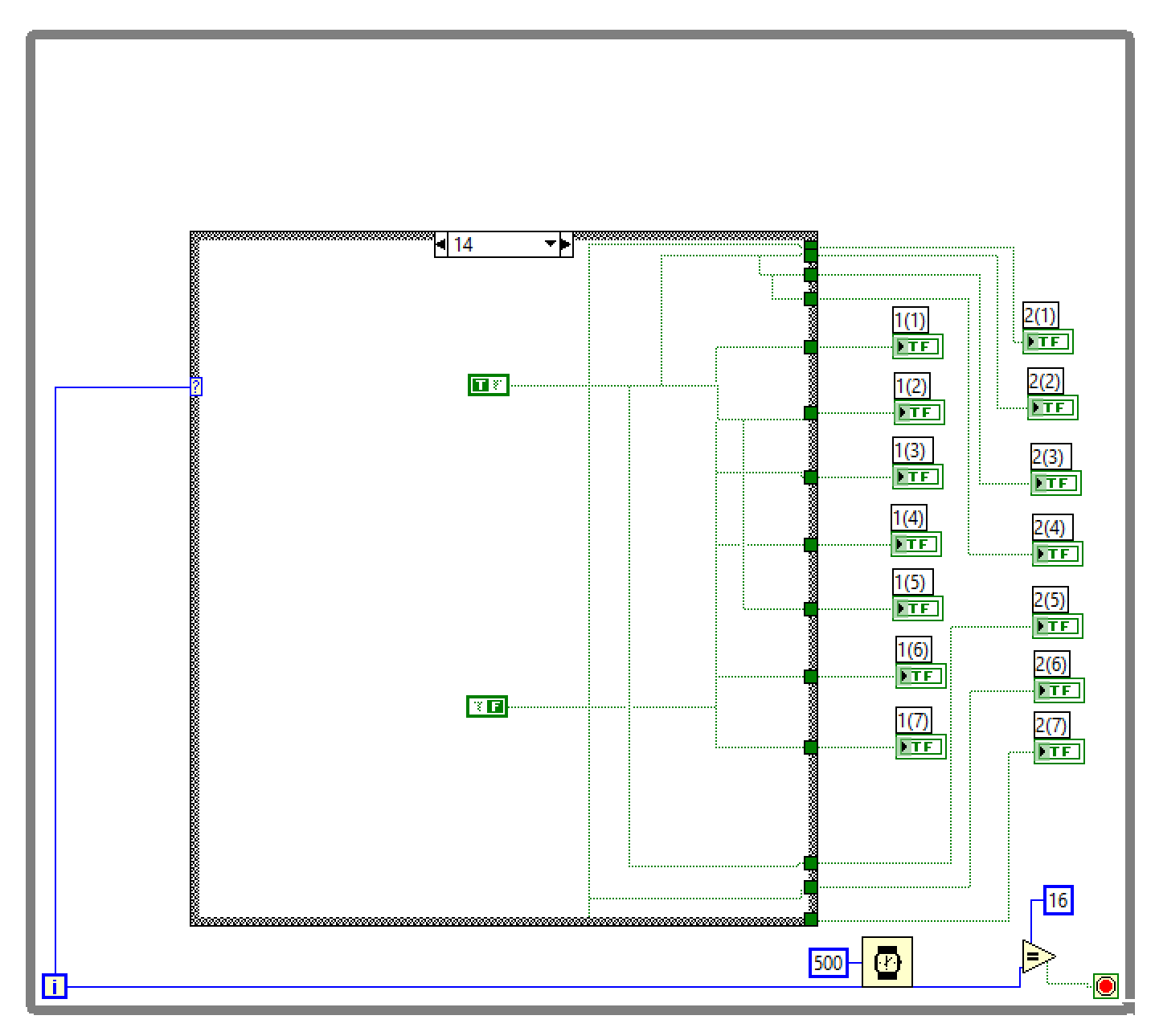


Рисунок. 46 – Программа *4.vi*(14 состояние)

На данном рисунке представлено 14 состояние из 16.

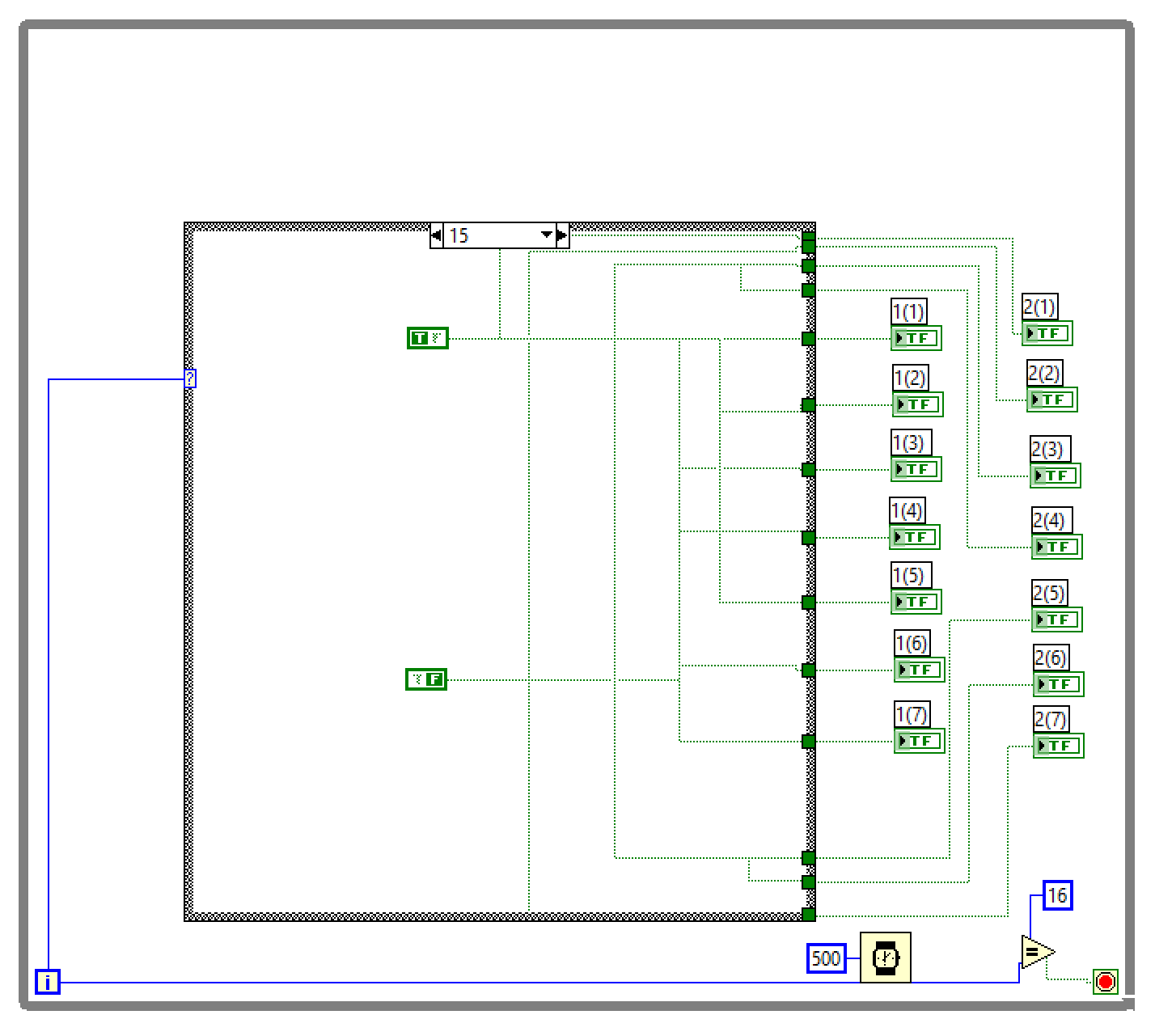


Рисунок. 47 – Программа *4.vi*(15 состояние)

На данном рисунке представлено 15 состояние из 16.

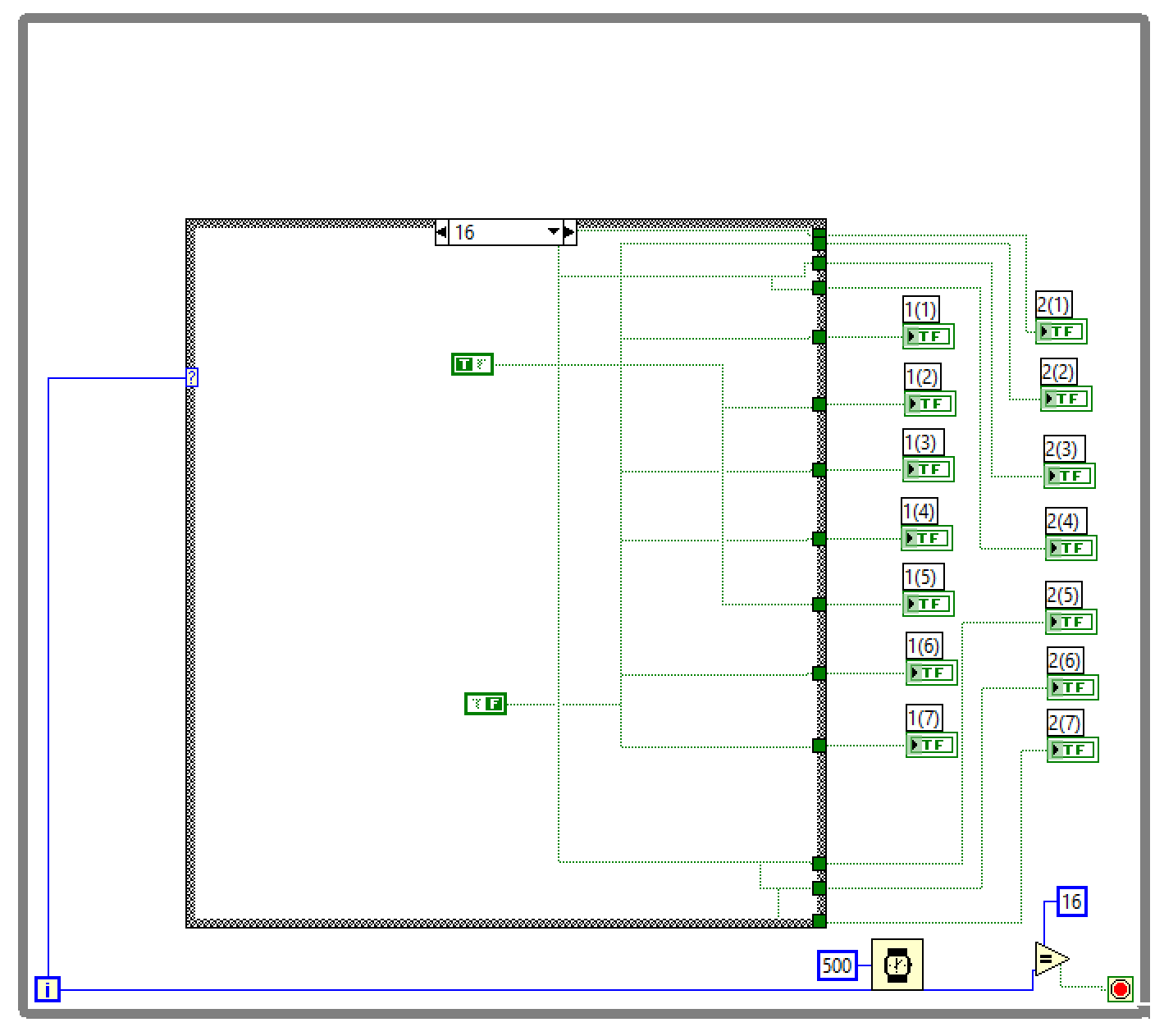


Рисунок. 48 – Программа *4.vi*(16 состояние)

На данном рисунке представлено 16 состояние из 16.

* 1. Результат программы:

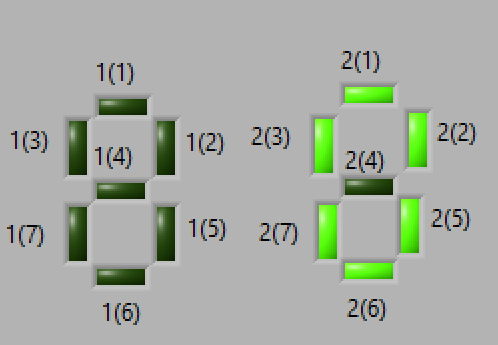


Рисунок. 49 – Начало работы программы

Начальное значение из 16. Следующее значение 1.

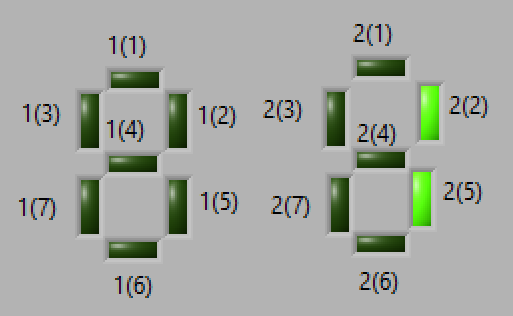


Рисунок. 50 – Первое состояние программы

1 состояние из 16. Следующее значение 2.

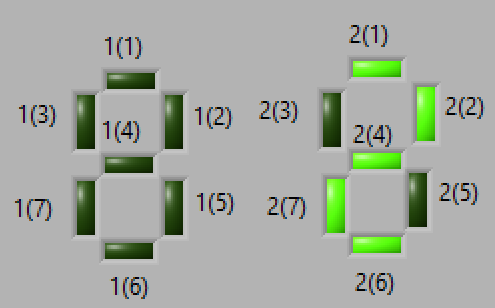


Рисунок. 51 – Второе состояние программы

2 состояние из 16. Следующее значение 3.

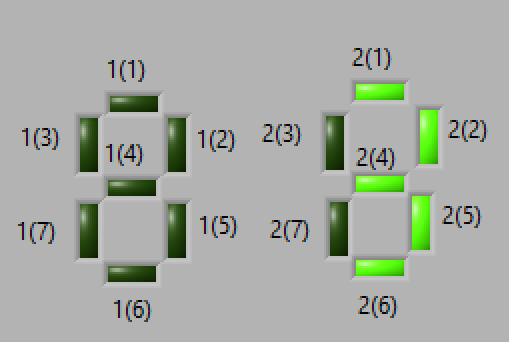


Рисунок. 52 – Третье состояние программы

3 состояние из 16. Следующее значение 4.

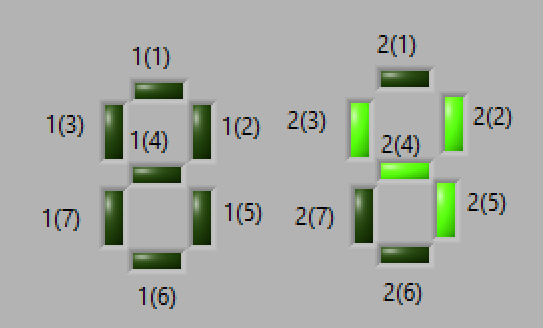


Рисунок. 53 – Четвертое состояние программы

4 состояние из 16. Следующее значение 5.

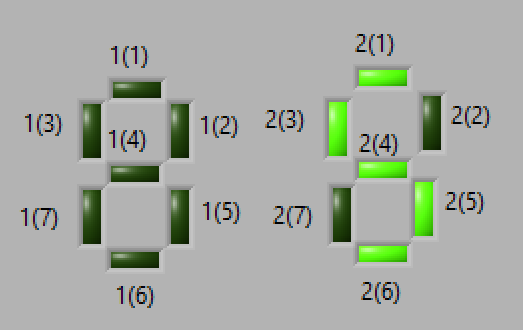


Рисунок. 54 – Пятое состояние программы

5 состояние из 16. Следующее значение 6.

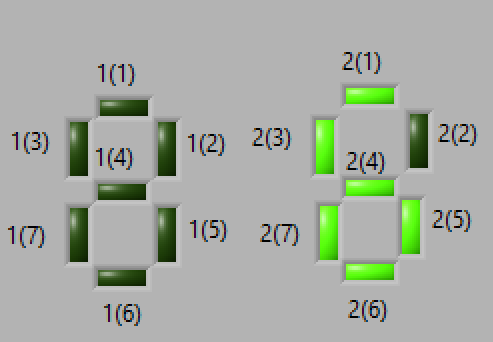


Рисунок. 55 – Шестое состояние программы

6 состояние из 16. Следующее значение 7.

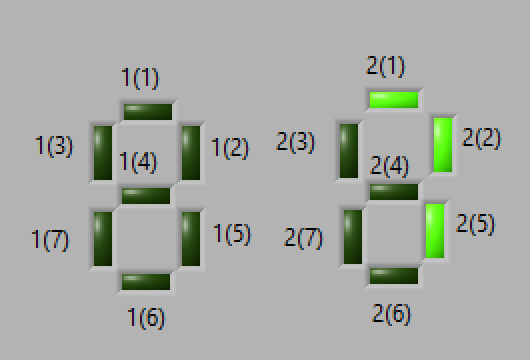


Рисунок. 56 – Седьмое состояние программы

7 состояние из 16. Следующее значение 8.

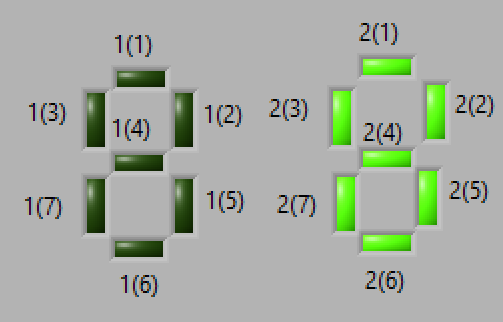


Рисунок. 57 – Восьмое состояние программы

8 состояние из 16. Следующее значение 9.

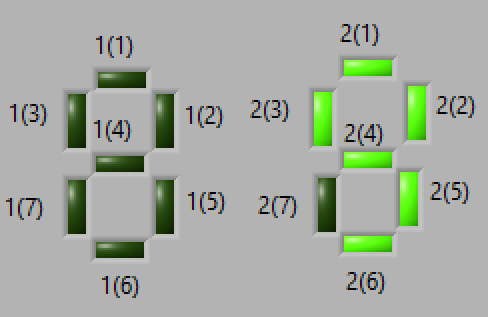


Рисунок. 58 – Девятое состояние программы

9 состояние из 16. Следующее значение 10.

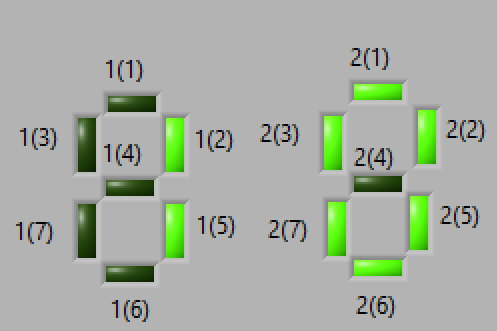


Рисунок. 59 – Десятое состояние программы

10 состояние из 16. Следующее значение 11.

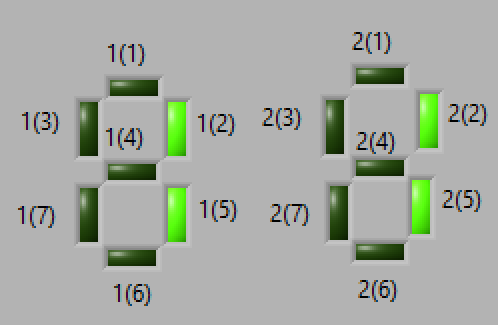


Рисунок. 60 – Одиннадцатое состояние программы

11 состояние из 16. Следующее значение 12.

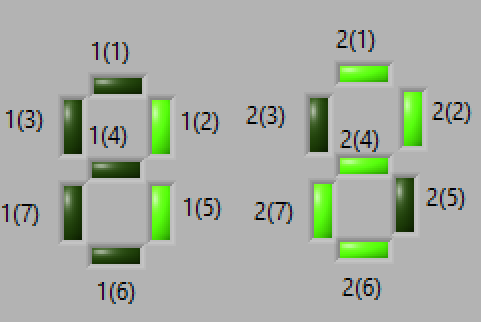


Рисунок. 61 – Двенадцатое состояние программы

12 состояние из 16. Следующее значение 13.

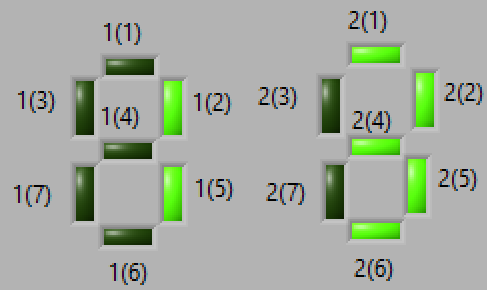


Рисунок. 62 – Тринадцатое состояние программы

13 состояние из 16. Следующее значение 14.

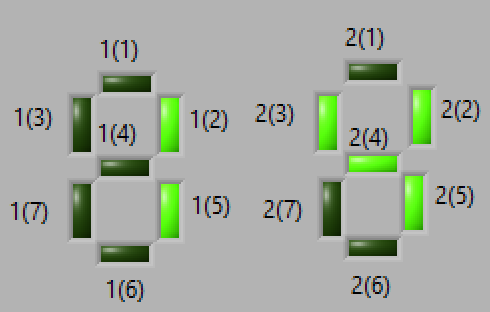


Рисунок. 63 – Четырнадцатое состояние программы

14 состояние из 16. Следующее значение 15.

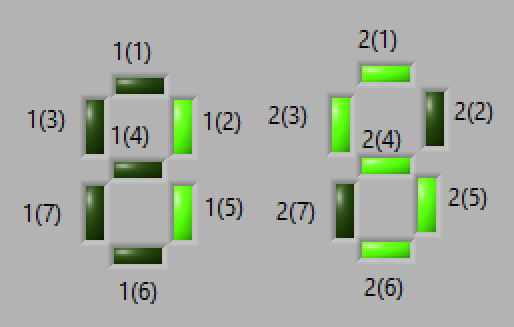


Рисунок. 64 – Пятнадцатое состояние программы

15 состояние из 16. Следующее значение 16.

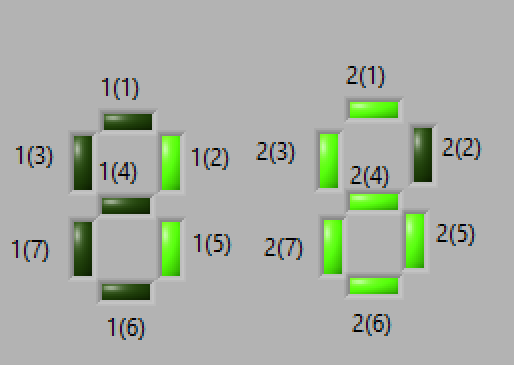


Рисунок. 65 – Последнее состояние программы

16 состояние из 16. Конец работы программы.

* 1. Блок-схема:

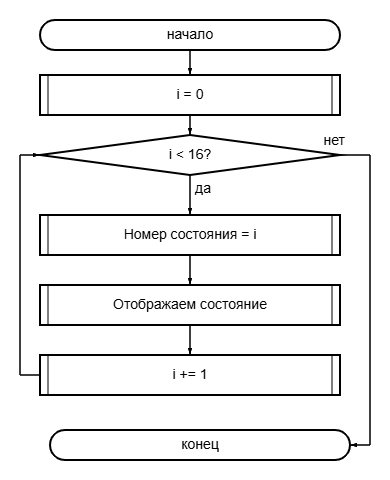


Рис. 66 – Блок-схема

* 1. Вывод:

Функционал программы *LabView* позволяет смоделировать работу счетчиков, светофоров и других устройств.