**Пример разработки виртуального прибора, содержащего датчик псевдослучайных чисел в циклической структуре:**

Простейший случай реализации виртуального прибора, моделирующего игру «Угадай число» (общая часть работы), состоит только из двух элементов, размещённых на передней панели: числовых контроллера и индикатора (Рисунок 1).

Контроллер задаёт число, которое должно быть отгадано, индикатор отвечает количеством попыток, потребовавшихся персональному компьютеру для угадывания числа. Важно помнить о необходимости ввода значимых имён для используемых интерфейсных элементов.

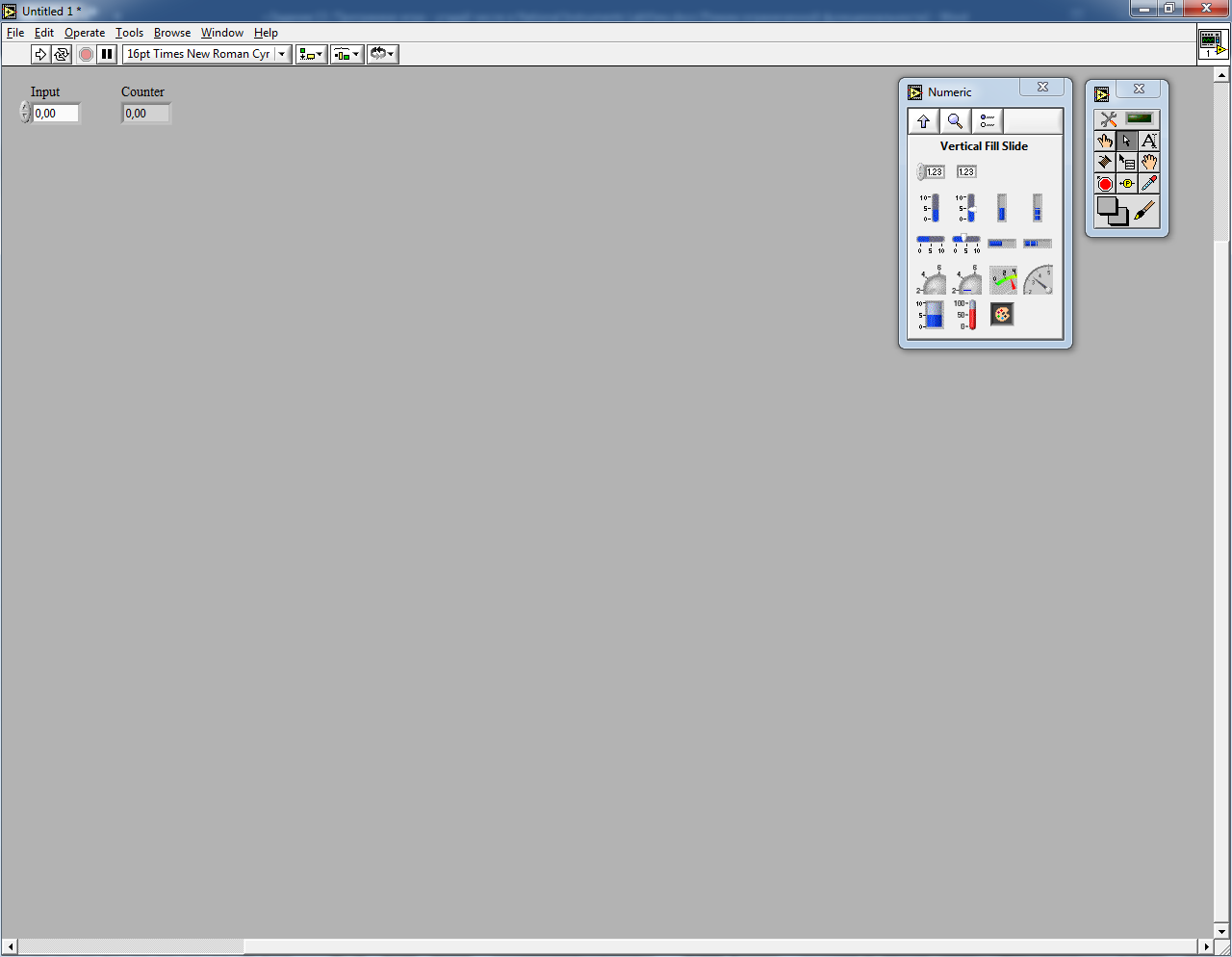


Рисунок 1 – Подготовка передней панели к созданию виртуального прибора, моделирующего игру «Угадай число»

Из описания следует, что рассматриваются целые числа. Если бы игра происходила между двумя людьми, то область допустимых значений была бы ограничена натуральными числами, однако данная игра моделируется на компьютере, и в учебных целях через неё разумно отработать навыки масштабирования псевдослучайной величины, распределённой от «0» до «1». В связи с этим не следует ограничиваться областью натуральных чисел, а нужно расширить её до области целых чисел. Поскольку использоваться должны целые числа, а не вещественные, то элементы, размещённые на передней панели, в своём исходном представлении для поставленной задачи обладают существенной избыточностью – они должны быть перенастроены с вещественного диапазона значений на один их целых. Делается эта настройка через контекстное меню уже размещённых на передней панели численных интерфейсных компонентов в разделе «Представление» (*Representation*). Необходимо от умолчания в виде вещественного диапазона значений двойной точности (*DBL*) перейти к целому диапазону значений, например, шестнадцатиразрядному (*I16*) – 65536 значений (от -32768 до 32767). Хотя, с точки зрения экономии выделяемой памяти для решения поставленной задачи, достаточно использовать и восьмиразрядное целое (*I8*) – 256 (от -128 до 127). В зависимости от варианта решаемой задачи этот параметр можно варьировать. Изменение представления данных проиллюстрировано на Рисунке 2.

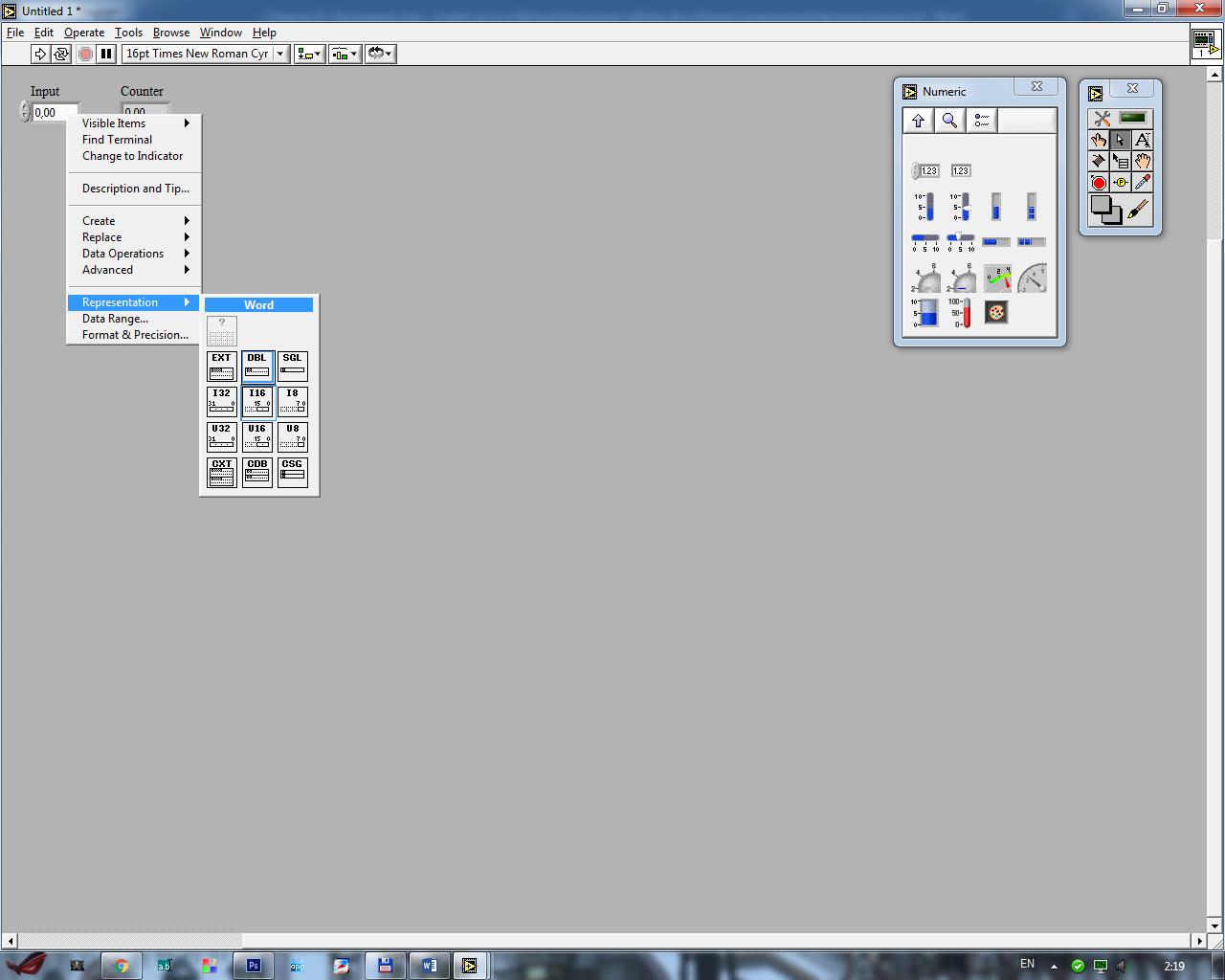


Рисунок 2 – Изменение типа входных данных с вещественного на целый

После изменения типа данных вещественная часть в интерфейсных элементах автоматически исключится. Данные будут представлены в соответствии с указанным типом данных – целыми (Рисунок 3).

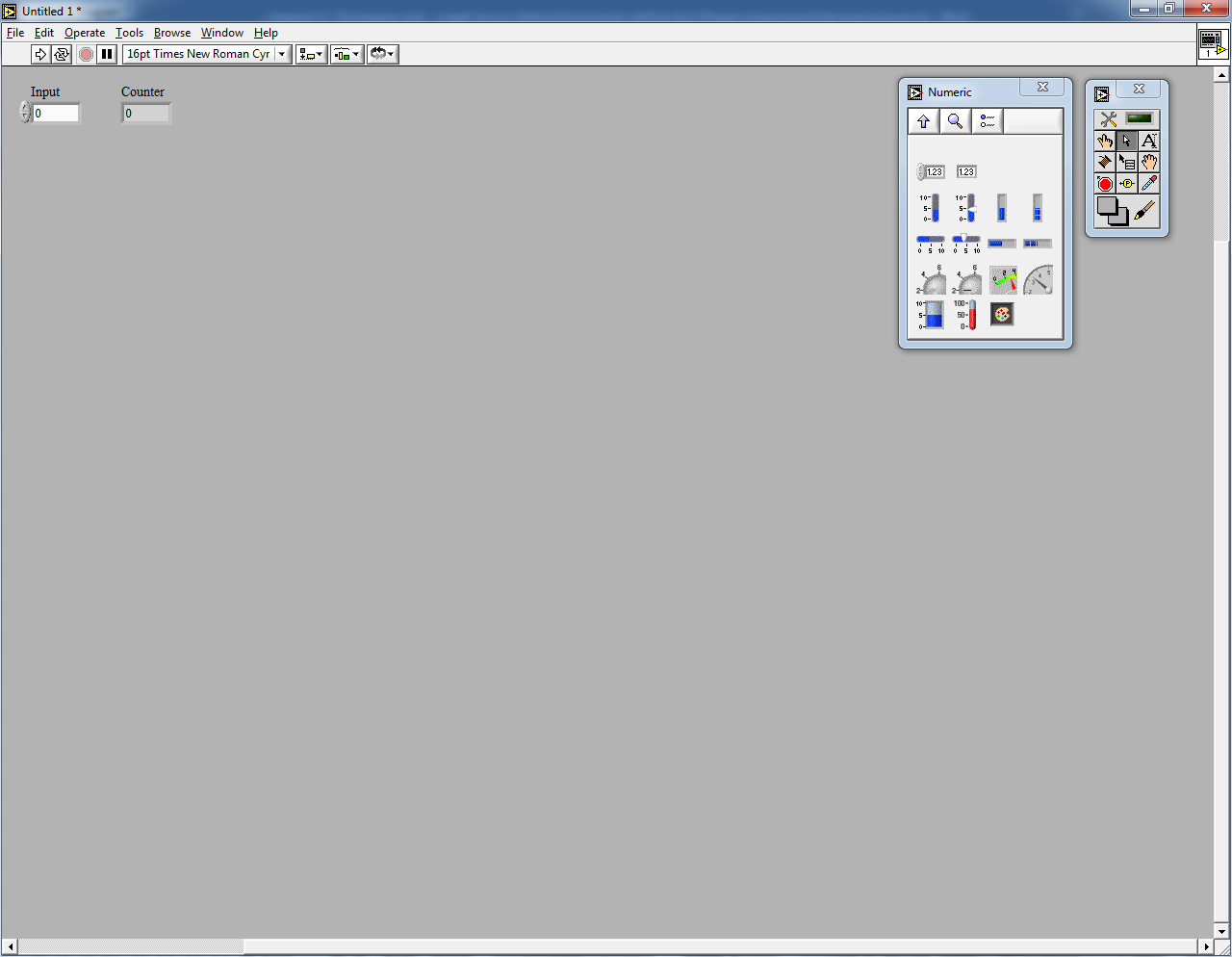


Рисунок 3 – Результат изменения входных и выходных данных с вещественного на целый

Тем временем на блок-диаграмме элементы тоже изменят своё состояние – окрасятся в соответствии назначенным типом данных. Целому типу, как показано на Рисунке 4, соответствует синий цвет.

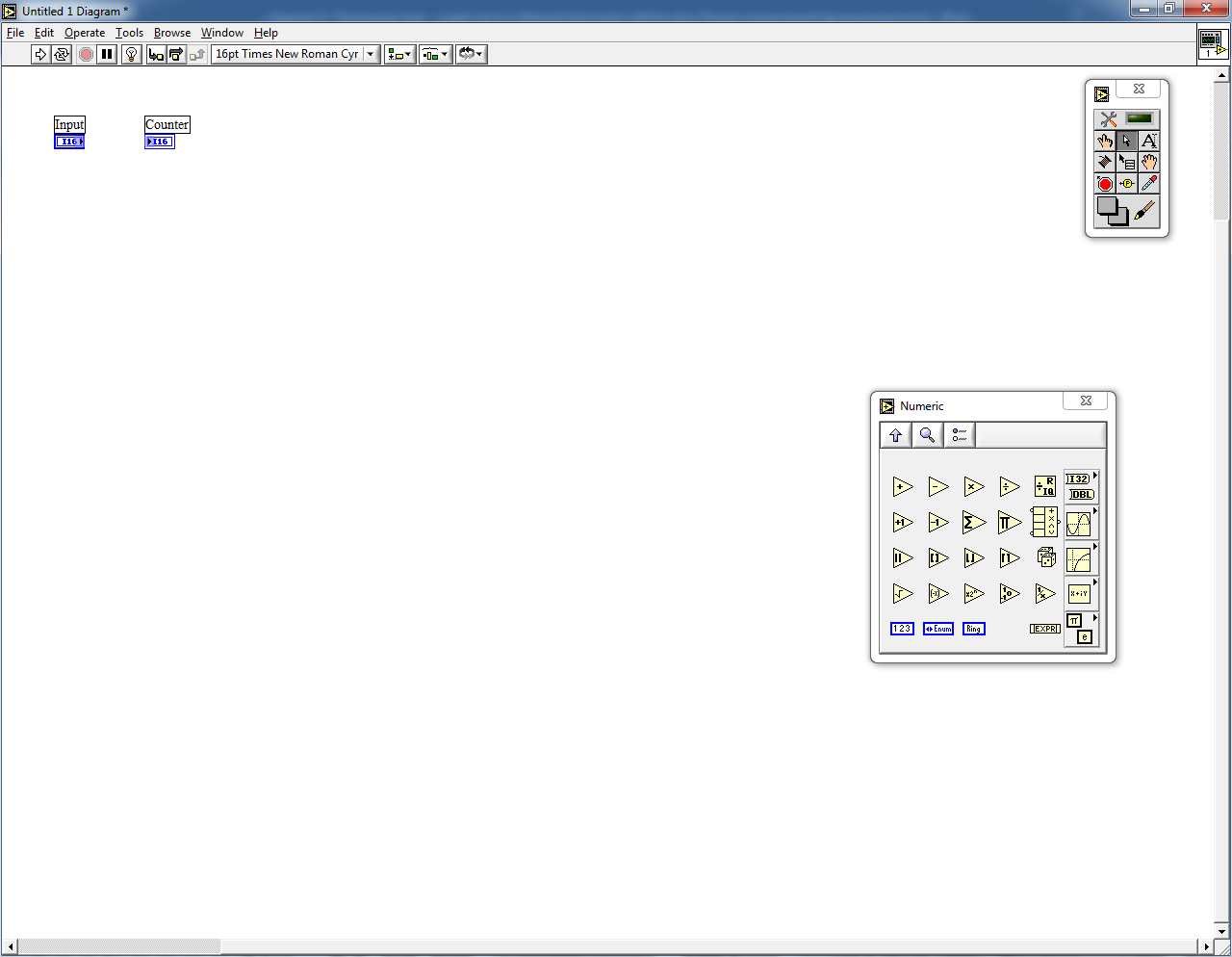


Рисунок 4 – Переход к блок-диаграмме создаваемого виртуального прибора. Отображение целочисленных элементов

Переместим индикатор, необходимый для вывода количества попыток, ниже контроллера со входным сигналом и изобразим между ними структуру цикла *While* (Рисунок 5).

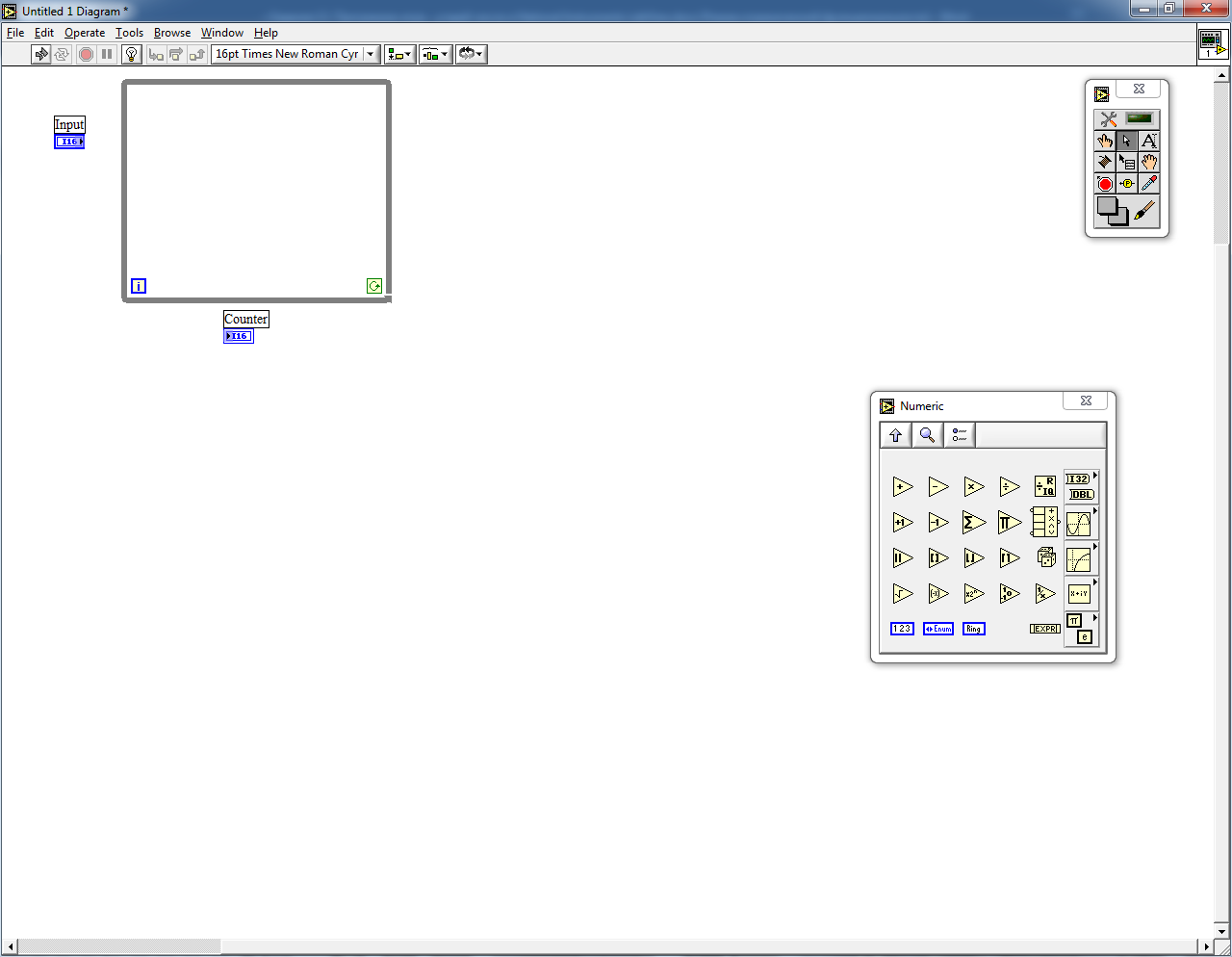


Рисунок 5 – Добавление структуры цикла *While* на блок-диаграмму

Напомним, что счётчик количества итераций цикла *i* начинается с нулевого значения, потому для вывода количества попыток этот счётчик необходимо инкрементировать (Рисунок 6).

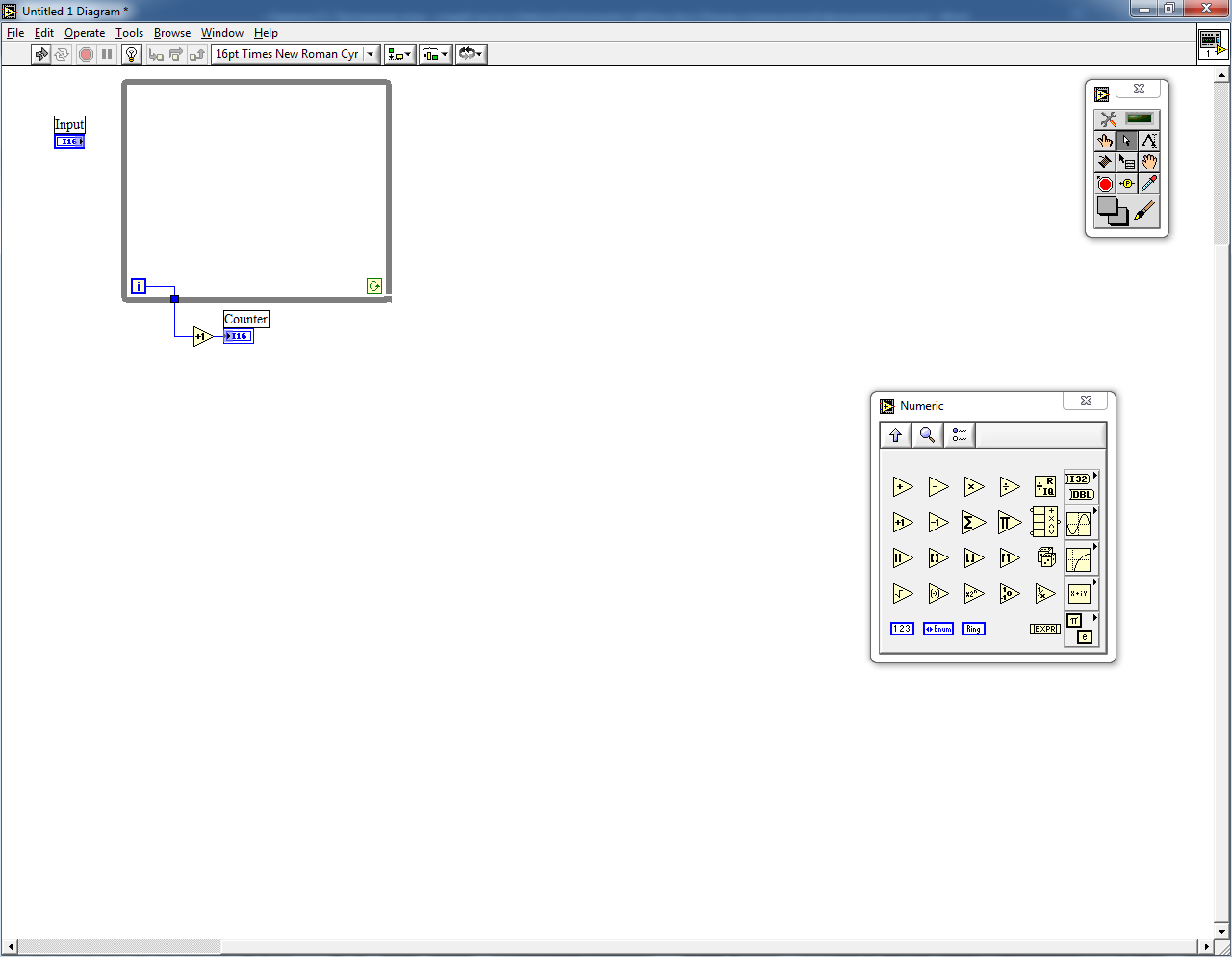


Рисунок 6 – Подключение параметра цикла *While* к выходным данным через операцию инкрементирования

Также удобно изменить режим циклической структуры *While* с продолжения по истине на остановку по истине (Рисунок 7), поскольку это диктуется правилами игры «Угадай число»: «Как только загаданное и отгаданное значения совпадут – игра завершится». Для завершения работы виртуального прибора цикл должен быть конечным.

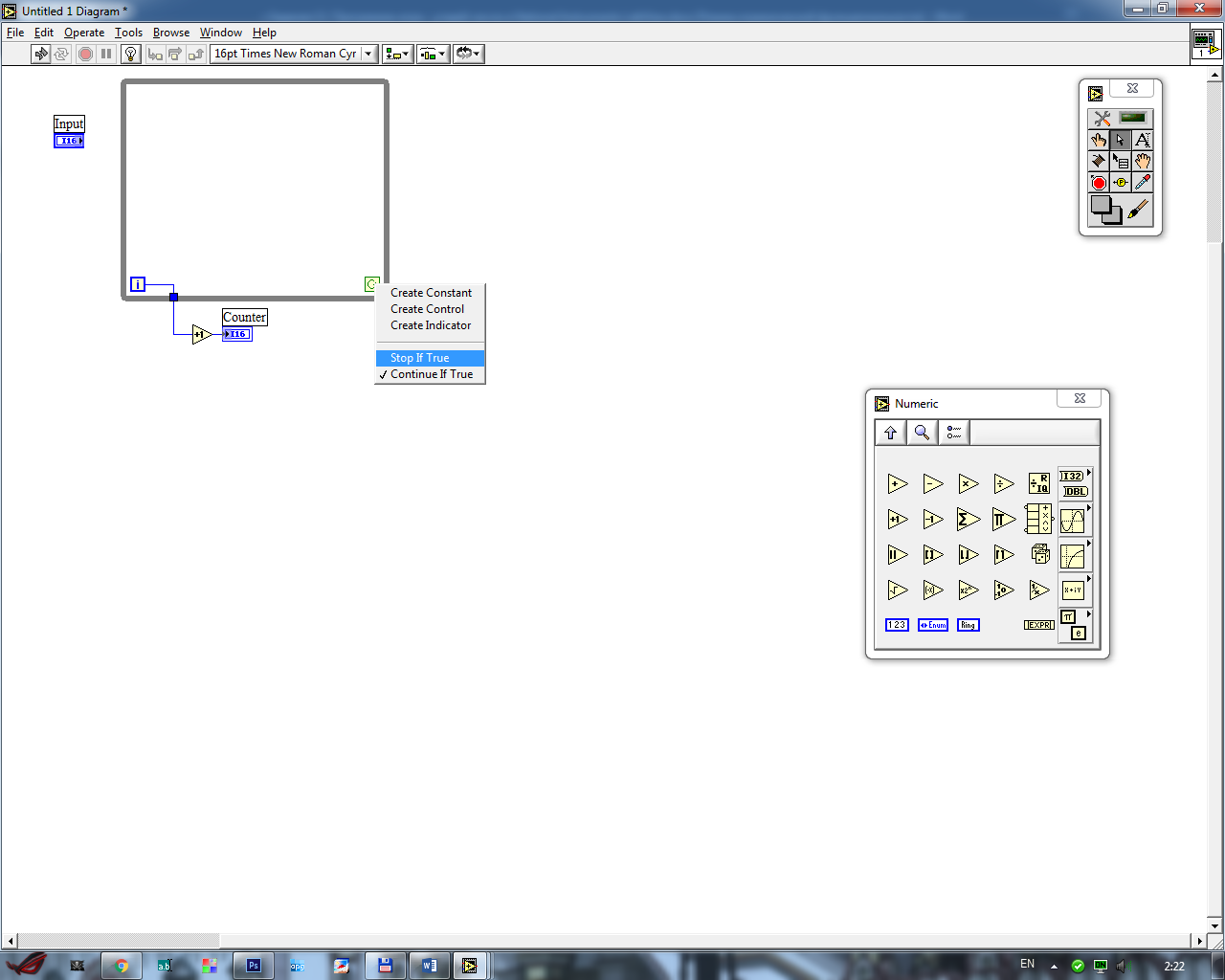


Рисунок 7 – Изменение режима работы цикла с «продолжения, если истина» (*Continue If True*) на «остановка, если истина» (*Stop If True*)

На следующем шаге программирования виртуального прибора необходимо организовать перебор значений случайным образом. В распоряжении составителя программного обеспечения имеется только вещественный датчик псевдослучайных числел, рапределённых в диапазоне от «0» до «1». Для того, чтобы диапазон значений получился шире, его необходимо смасштабировать и, в отдельных случаях, сместить. Для примера рассмотрим диапазон значений от «-10» (не включая) до «10» (включая).

Сначала выполним масштабирование. Пример содержит двадцать точек – возможных вариантов ответа. Так, перемножив значение на выходе датчика псевдослучайных чисел на количество точек (20), мы выполнили масштабирование на интересующий дипазон, однако, реализации псевдослучайной величины будут относиться к диапазону от «0» до «20» (Рисунок 8).

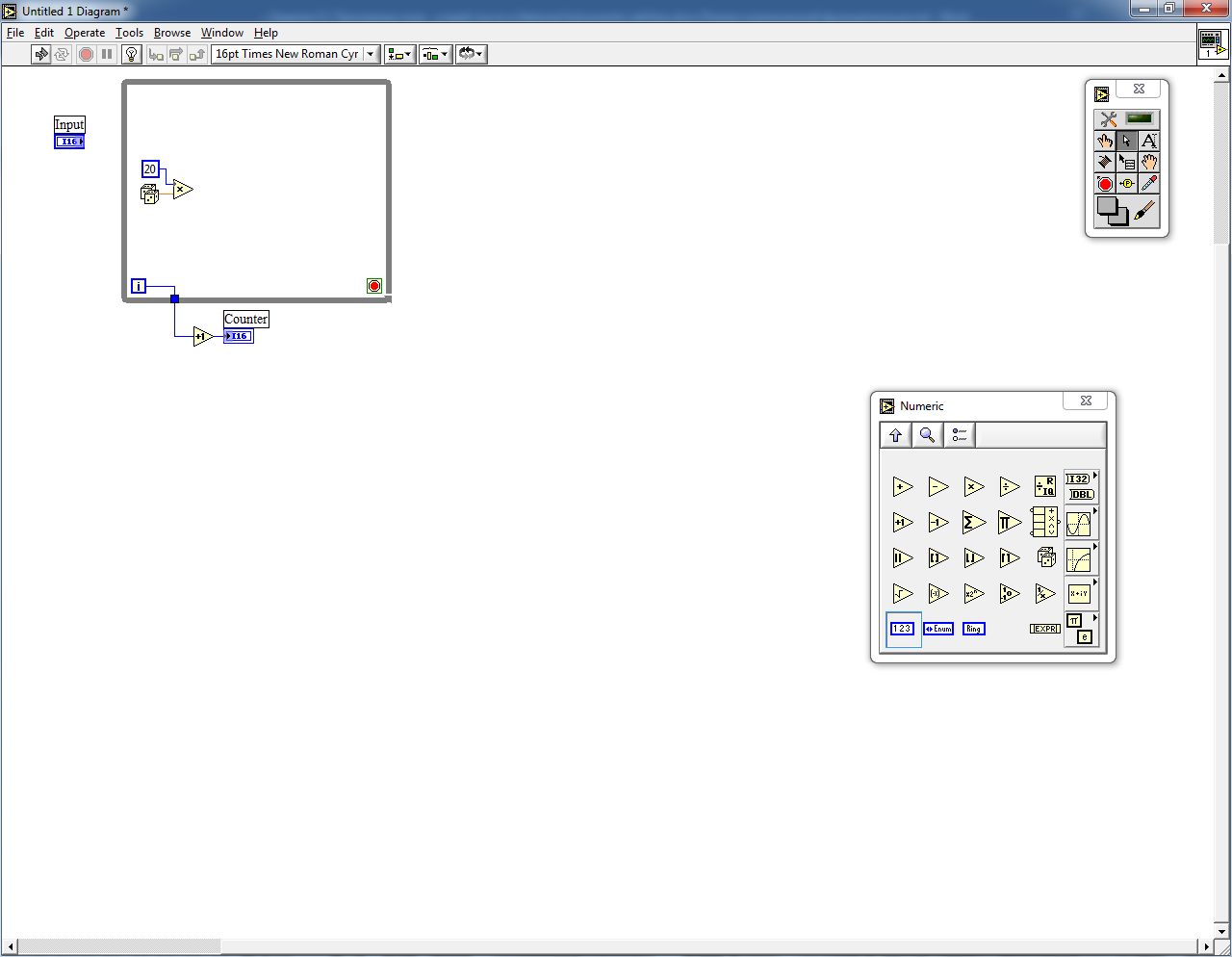


Рисунок 8 – Масштабирование с диапазона 0..1 до диапазона 0..20

При этом на выходе всё равно останется вещественное число, которое при сравнении с целым не даст нужного результата сравнения. Вероятность попадания в каждую конкретную вещественную точку равна к нулю, потому необходимо отойти от вещественных значений. Округлим полученный результат до целого в большую сторону (Рисунок 9).

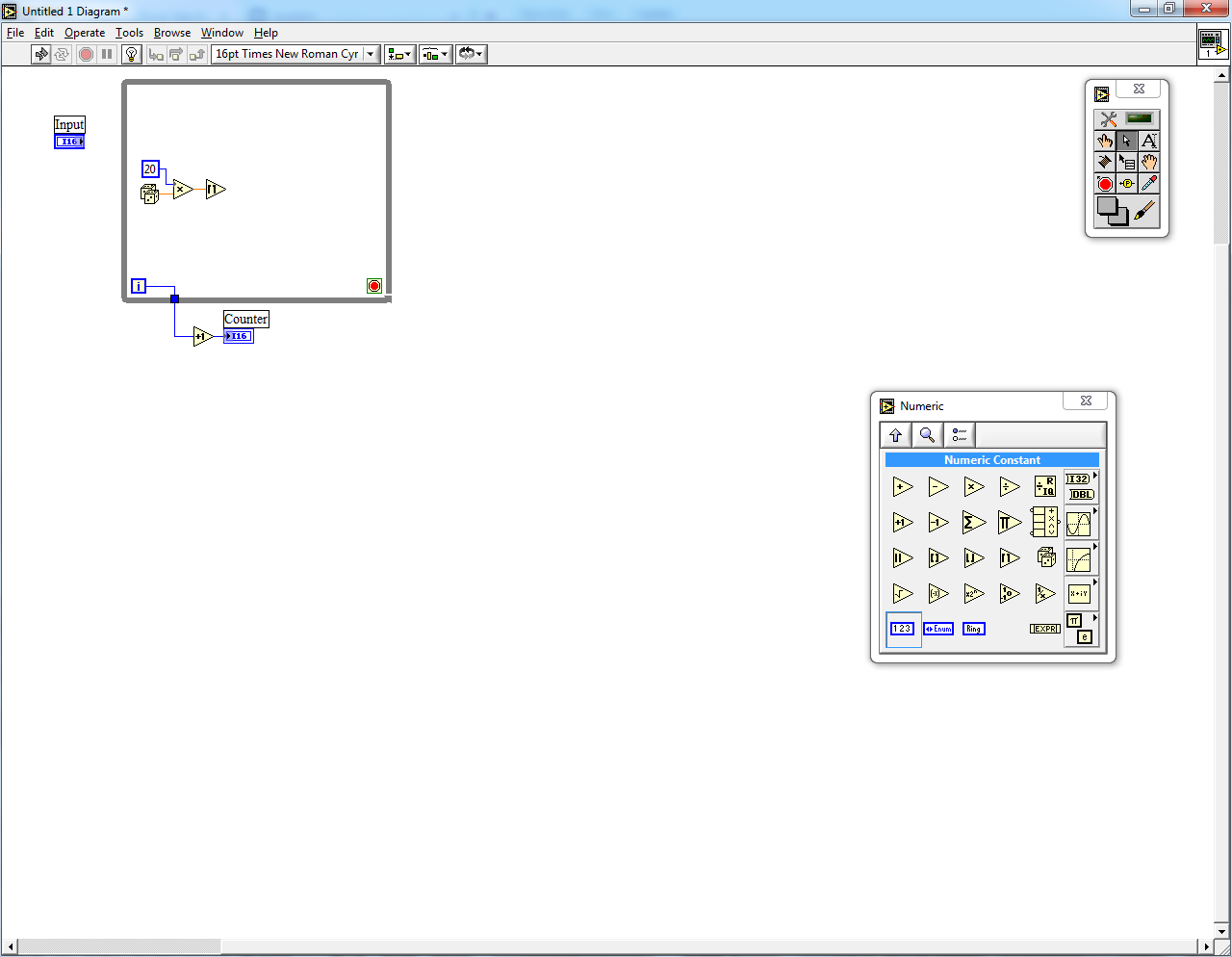


Рисунок 9 – Округление масштабированного значения в сторону большего целого

После этого можно организовать смещение в область отрицательных значений, для чего необходимо (Рисунок 10):

1. Разделить надвое величину диапазона;

2. Округлить результат в меньшую сторону (для нечёных величин диапазонов). Полученный результат является величиной смещения;

3. Из масштабированного значения в диапазоне вычесть полученное значение для смещения.

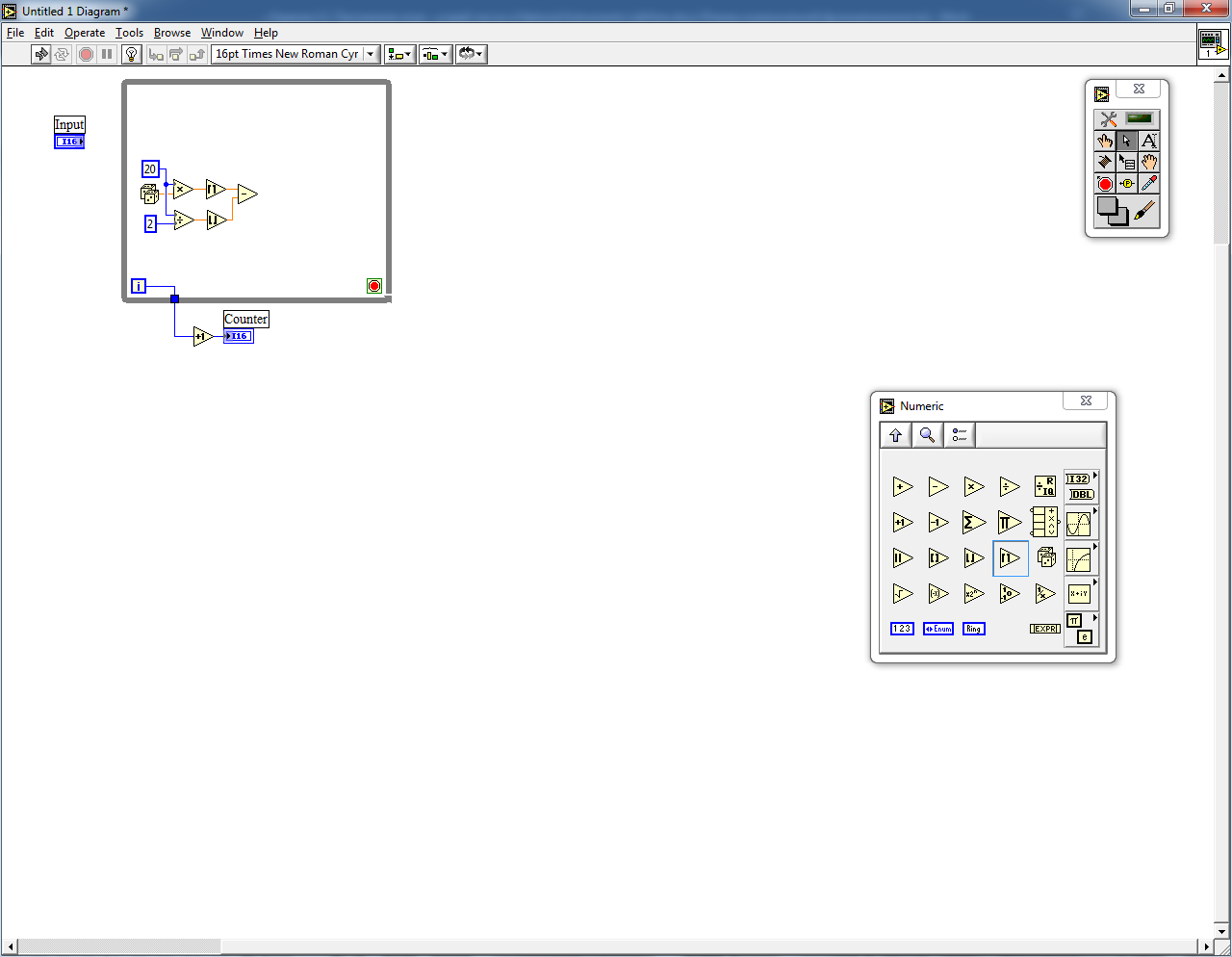


Рисунок 10 – Смещение масштабированного значения на половину диапазона в отрицательную область

Значение, вычисленное с учётом смещения, и есть та величина, которую предлагает компьютер в качестве варианта ответа – её необходимо сравнить со значением, введённым пользователем. В том случае, если они совпадут, элемент сравнения выдаст на элемент управления циклической конструкции *While* сигнал со значением «истина» (*true*), что для режима «остановка по истине» является условием прекращения работы цикла. К тому моменту счётчик количества итераций цикла насчитает определённое значение, которое с учётом инкрементирования будет выдавать количество предпринятых попыток угадывания заданного числа на индикатор (Рисунок 11).

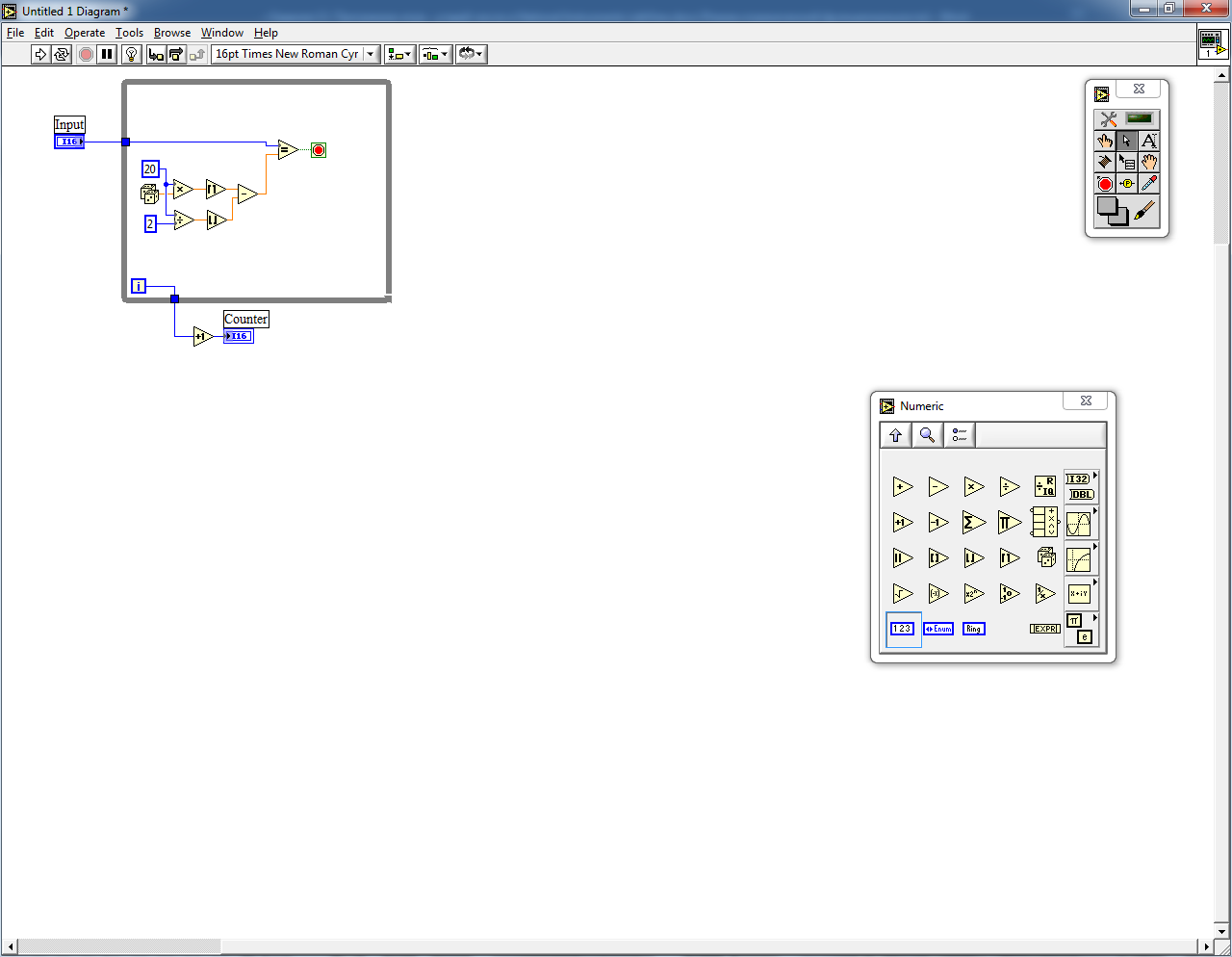


Рисунок 11 – Сравнение значений: поданного на вход и полученного масштабированным датчиком случайных чисел

Далее на Рисунках 12 и 13 представлены тестовые примеры работы виртуального прибора на границах рассматриваемого диапазона. Получены ненулевые значения количества попыток в обоих случаях, что подтверждает адекватность составленного программного обеспечения и алгоритма его работы.

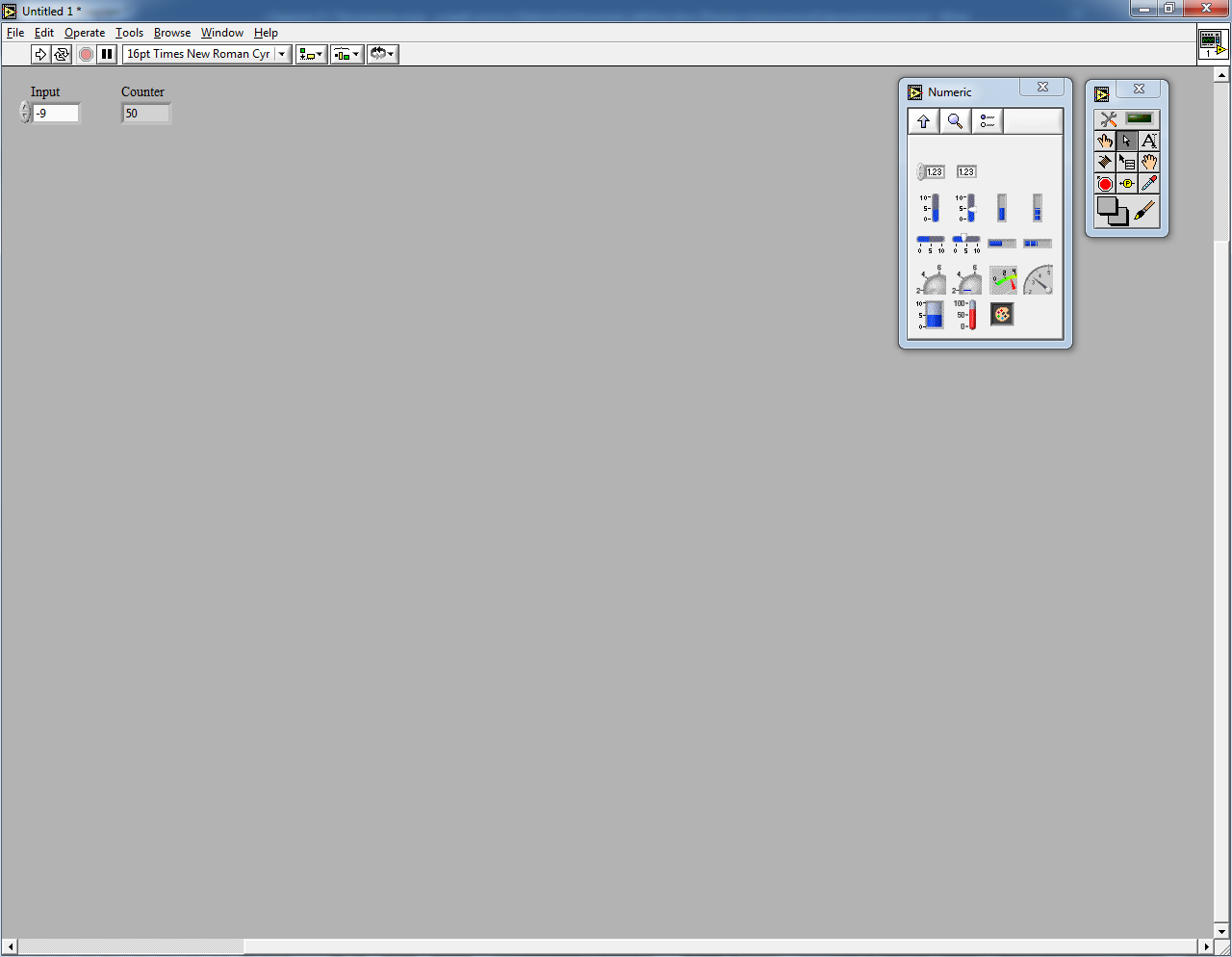


Рисунок 12 – Результат тестирования виртуального прибора по нижней границе диапазона

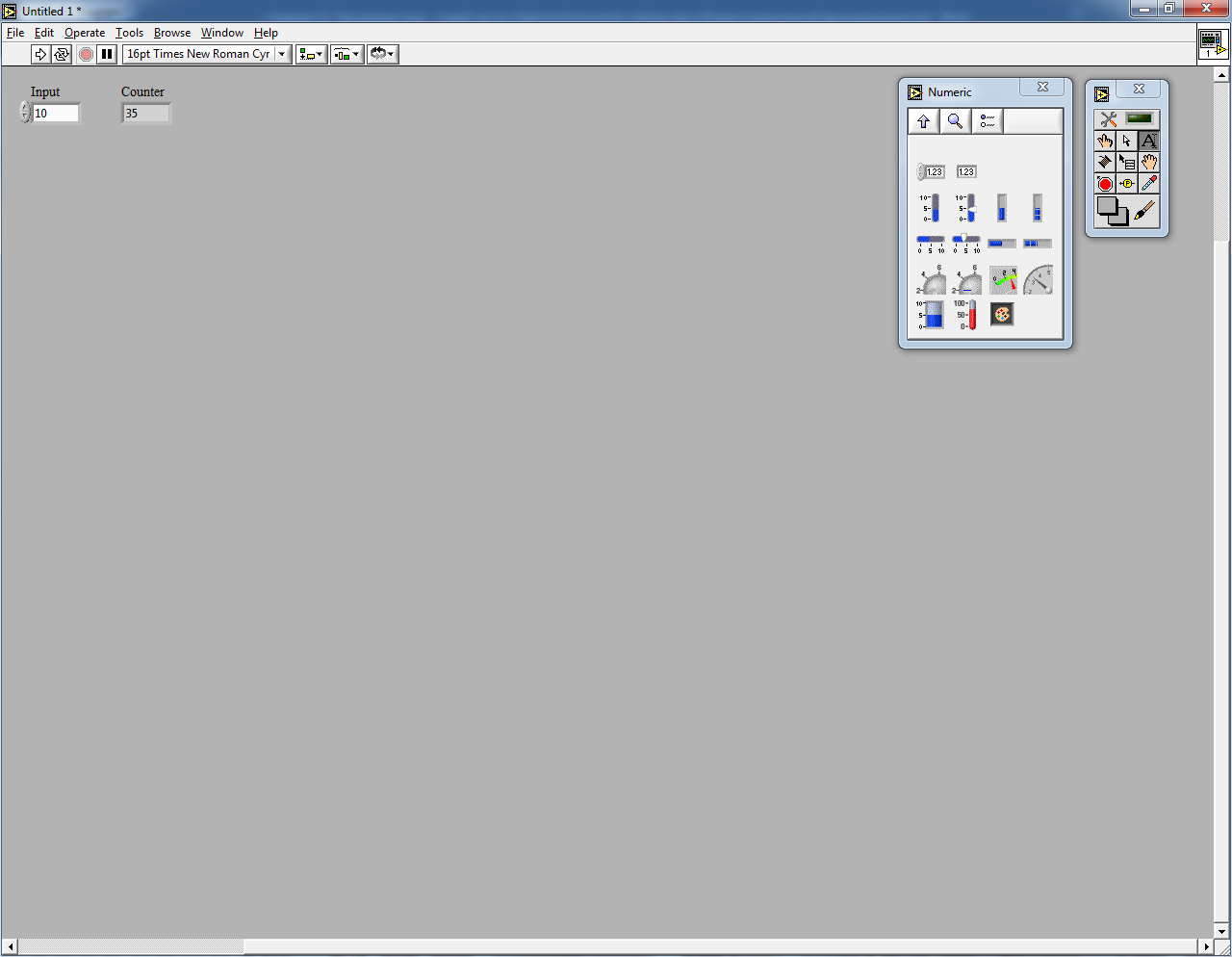


Рисунок 13 – Результат тестирования виртуального прибора по верхней границе диапазона

Не стоит забывать о минимизации схемы виртуального прибора, размещаемой на блок-диаграмме, по итогам проделанной работы. Одна из возможных реализации компактной схемы игры «Угадай число» представлена на Рисунке 14.

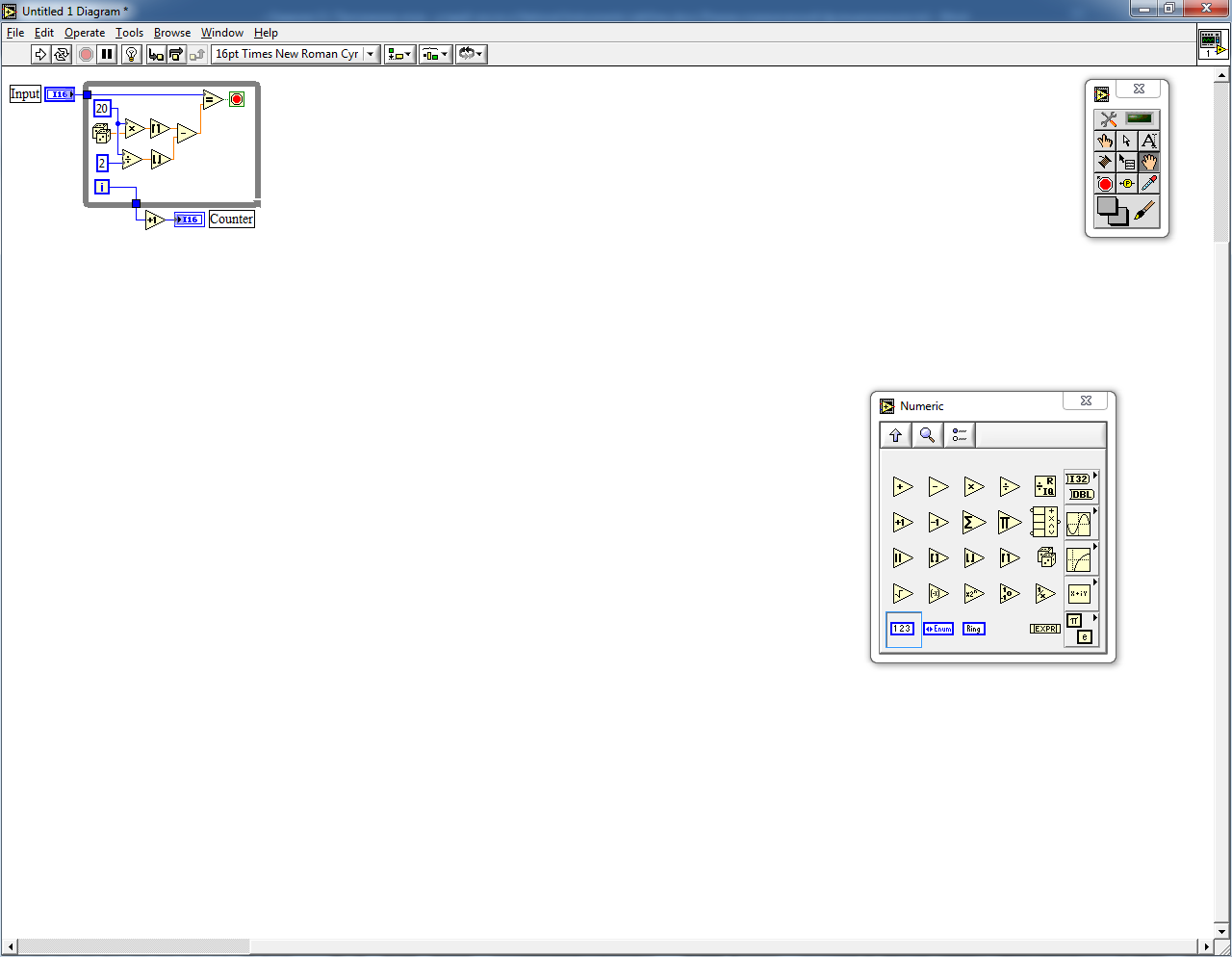


Рисунок 14 – Результат минимизации кода, размещённого на блок-диаграмме

----------------------------------------------------------------------------------------------------

**Пример модификации структуры игры «Угадай число» с целью учёта промежуточных (не угаданных) значений:**

Индивидуальная часть работы предполагает изучение обучающимися с применением на практике более широкого набора элементной базы *National Instruments LabView*, по сравнению с тем, который прописан в качестве базового минимума для выполнения общей части работы по составлению виртуального прибора, моделирующего игру «Угадай число»: массивов, строк, функций, процедур, графиков функциональных зависимостей.

До настоящего момента задания Учебной практики должны были помочь обучающимся в достаточном объёме освоить, практически, каждый из упомянутых выше разделов за исключением одного – раздела строк.

Строки ни в одной из ранее предложенных к выполнению работ не разбирались, потому особое внимание в рамках примера выполнения индивидуальной части работы по составлению виртуального прибора, моделирующего игру «Угадай число», уделяется именно этому типу данных.

Строковые элементы собраны в объединённом разделе с другим типом данных – с путями к файлам. В широком смысле для программирования и пути, и строки – одно и то же, однако, в *National Instruments LabView* эти компоненты относятся к разным типам данных и маркируются, соответственно, различным цветом: строки – розовым, пути – бирюзовым.

От строкового раздела (Рисунок 15) для передней панели потребуется только строковый индикатор. В качестве входной информации строки в данной работе не используются, они применяются только для вывода данных – вывода результатов.

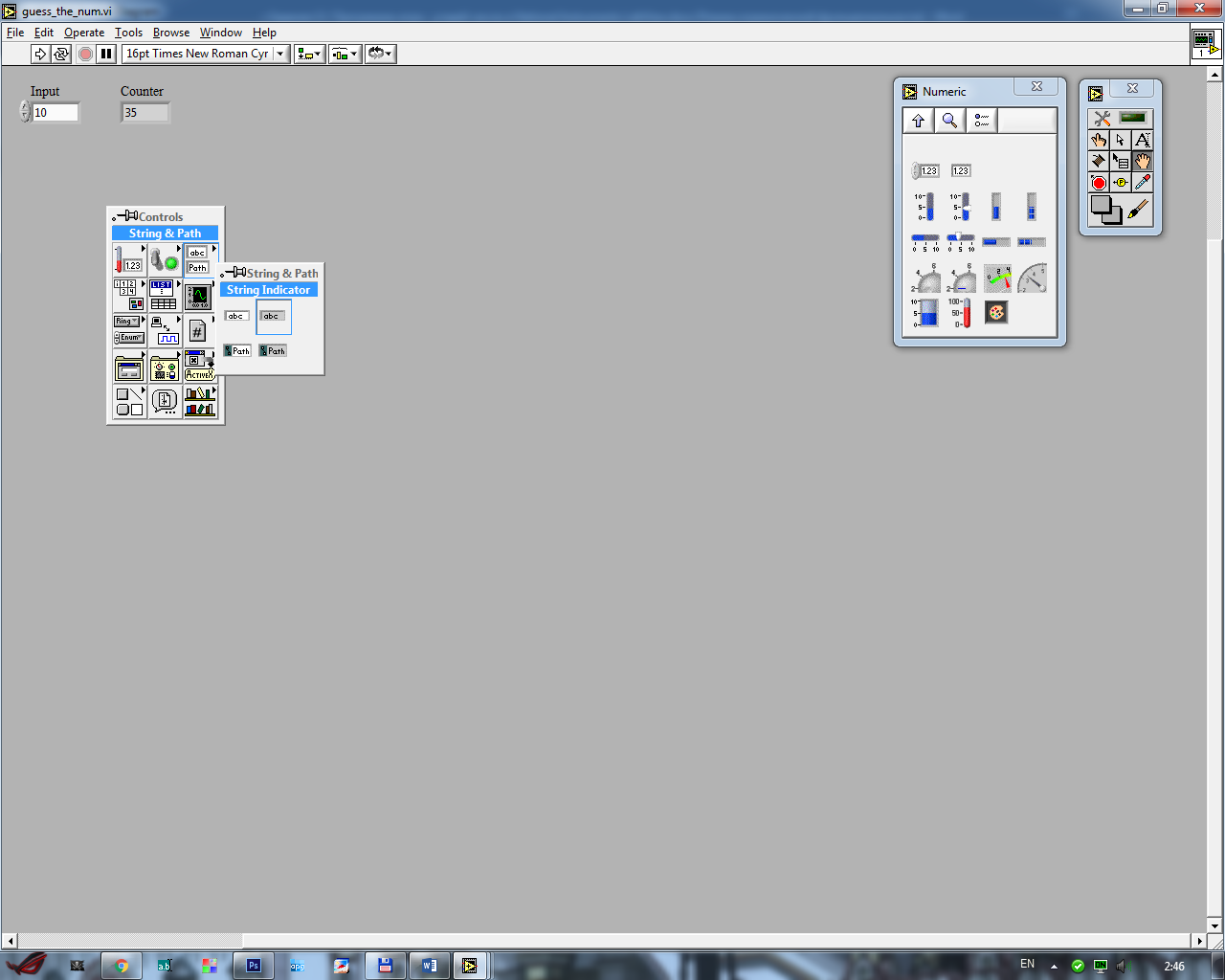


Рисунок 15 – Маркировка строкового индикатора (*String Indicator*) раздела строк и путей (*String & Path*) интерфейсных элементов управления

Разместим строковый индикатор на передней панели виртуального прибора, моделирующего игру «Угадай число» (Рисунок 16).

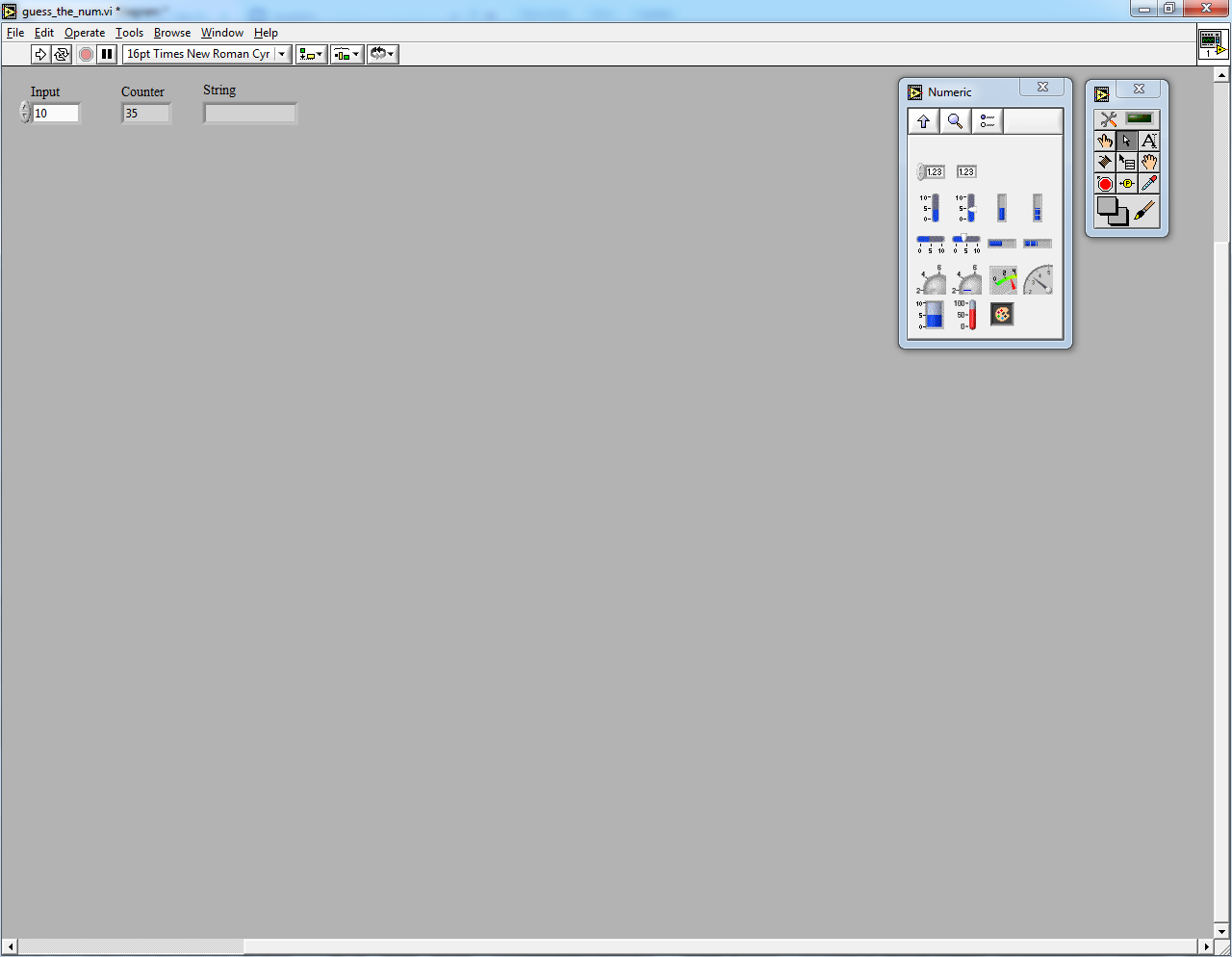


Рисунок 16 – Добавленный на переднюю панель строковый индикатор

Попыток, предпринятых персональным компьютером, может оказаться немало, потому сразу расширим область строкового индикатора под вывод большого количества символов (Рисунок 17). Делается это при помощи инструмента позиционирования / изменения размерности / выбора. Вместе с тем присвоим элементу значащее имя «Попытки» (*Attempts*).

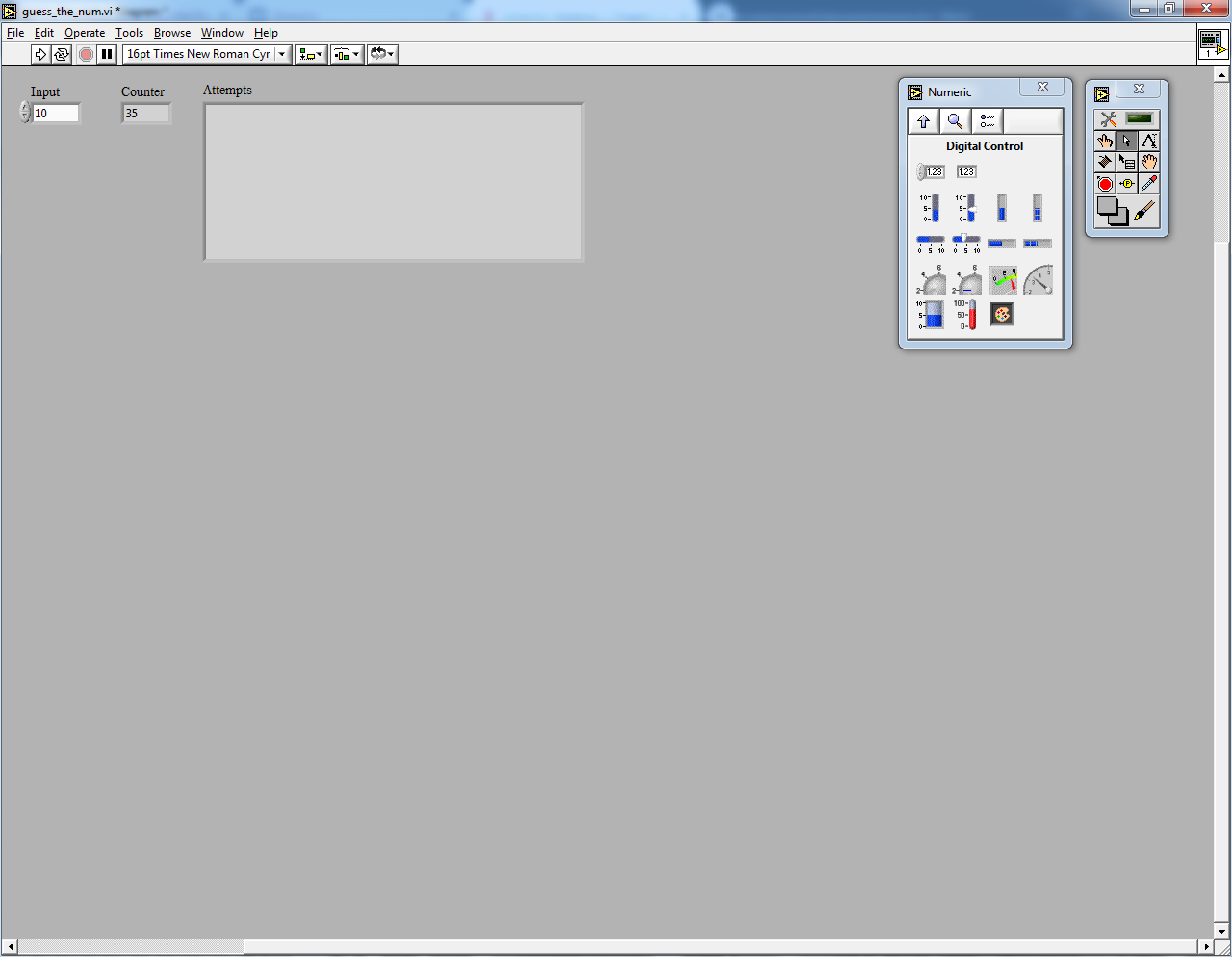


Рисунок 17 – Настройка строкового индикатора для вывода большого количества значений

Перейдём к блок-диаграмме. Итак, согласно задумке на каждой итерации цикла строка, содержащая сведения о перебираемых компьютером значениях, должна дополняться новыми сведениями. Так в структуру цикла необходимо включить сдвиговый регистр (*Add Shift Register*), призванный передавать значение, полученное на текущей итерации цикла, на следующую итерацию цикла. В качестве начального приближения для формирования строки на вход сдвигового регистра, ответственного за составление строки, необходимо подать пустую строку (строковая константа, расположенная в разделе строковых элементов и функций). Этот элемент (Рисунок 18) является одновременно и элементом сброса данных перед входом в цикл (очистка переменной), и элементом, задающим тип данных сдвигового регистра.

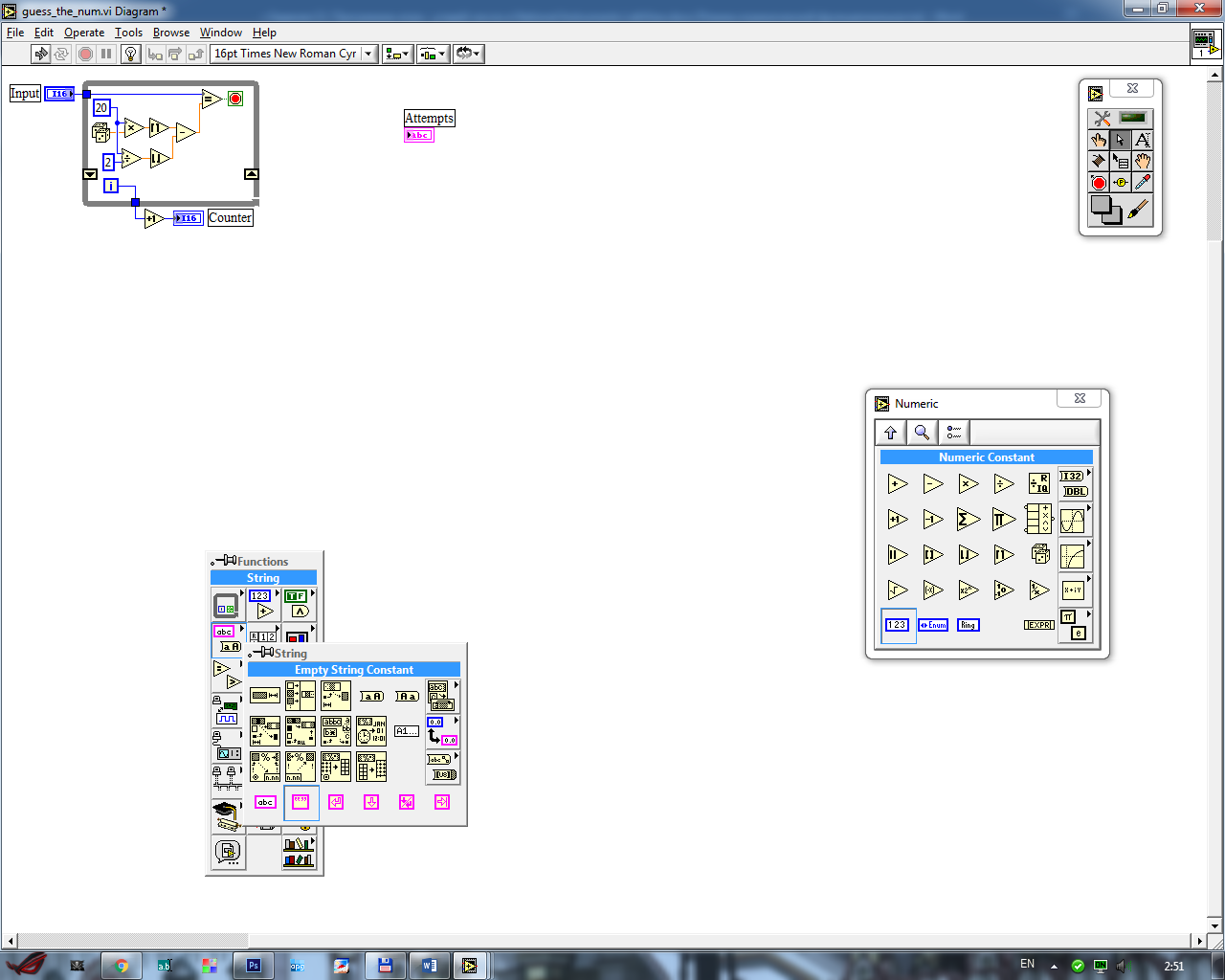


Рисунок 18 – Маркировка пустой строковой константы (*Empty String Constant*) раздела строк (*String*) функций, размещаемых на блок-диаграмме

Начальный этап модификации виртуального прибора «Угадай число» для записи значений, предложенных в качестве правильного ответа, представлен на Рисунке 19.

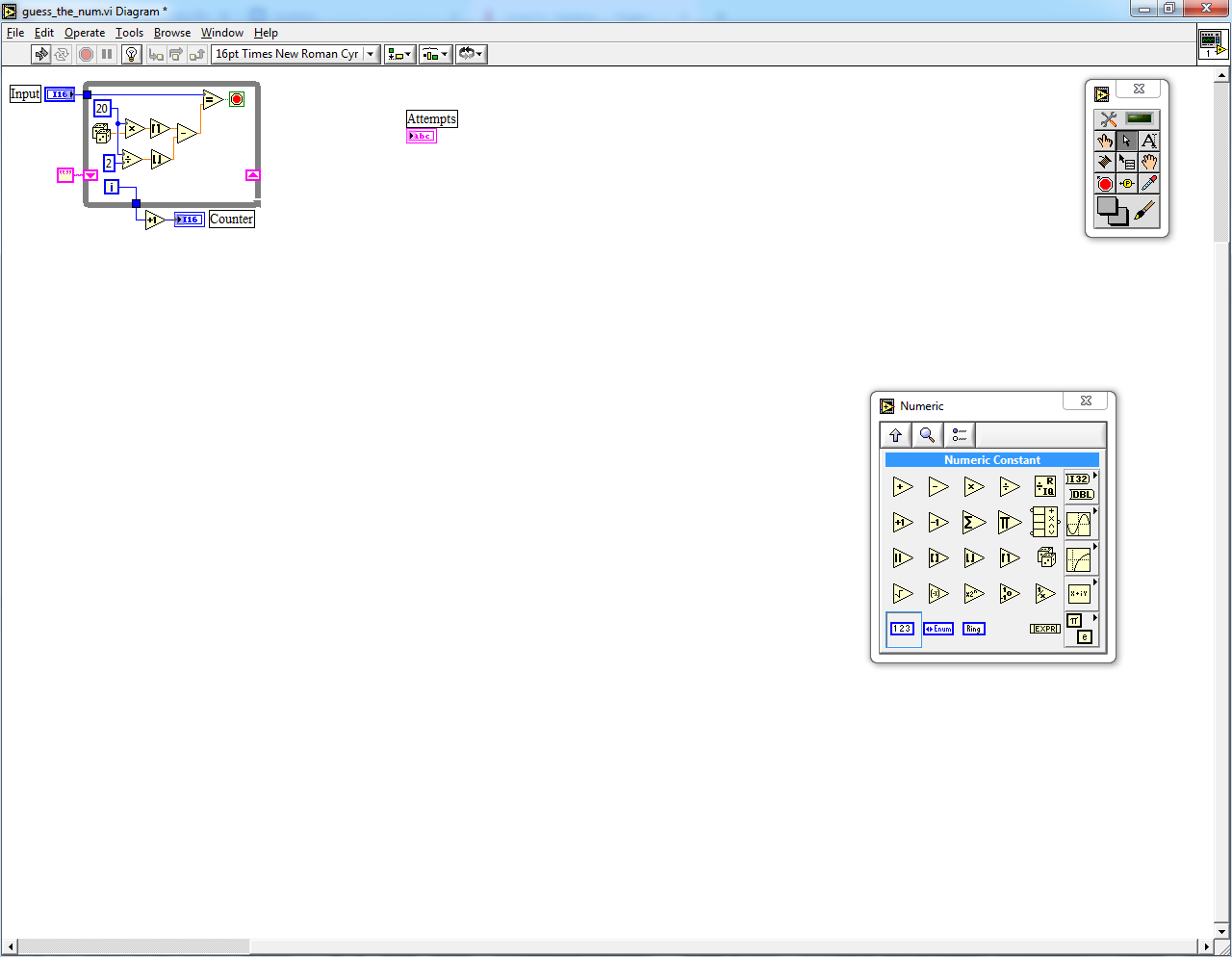


Рисунок 19 – Подключение пустой строковой константы на вход сдвигового регистра

Задача формирования строки в цикле идейно состоит в следующем: «Необходимо к тому, что было, дописать справа что-то новое». Таким образом, для слияния двух строк используется специальный оператор конкатенации (*Concatenate Strings*). Он располагается всё в том же разделе строковых элементов и функций (Рисунок 20).

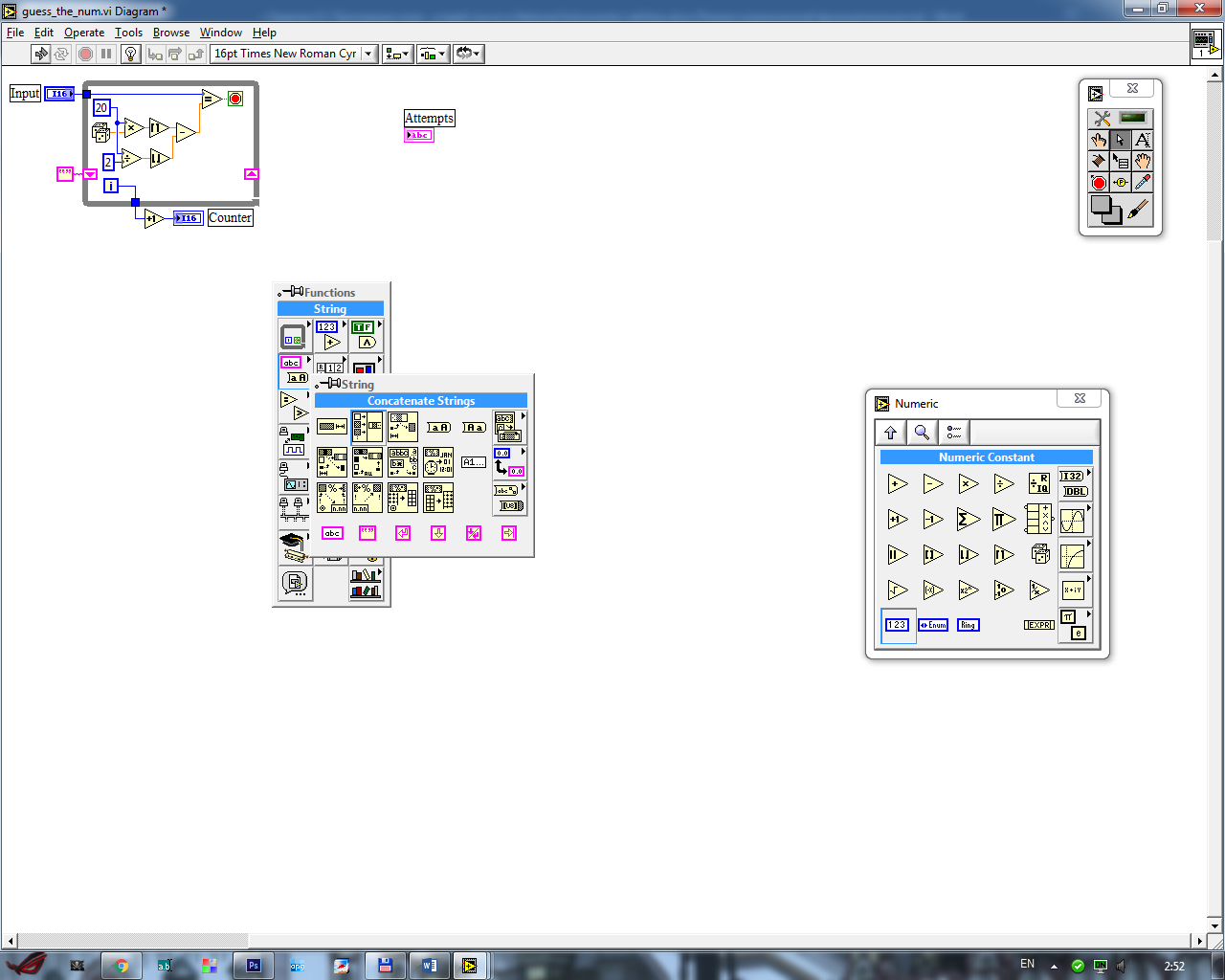


Рисунок 20 – Маркировка функции конкатенации строк (*Concatenate Strings*) раздела строк (*String*) функций, размещаемых на блок-диаграмме

На Рисунке 21 реализована лишь часть упомянутой идеи: «К тому, что было…». Тем не менее, эта строковая конструкция уже готова к тому, чтобы быть дополненной «чем-то новым». Пока «что-то новое» ещё не сформировано.

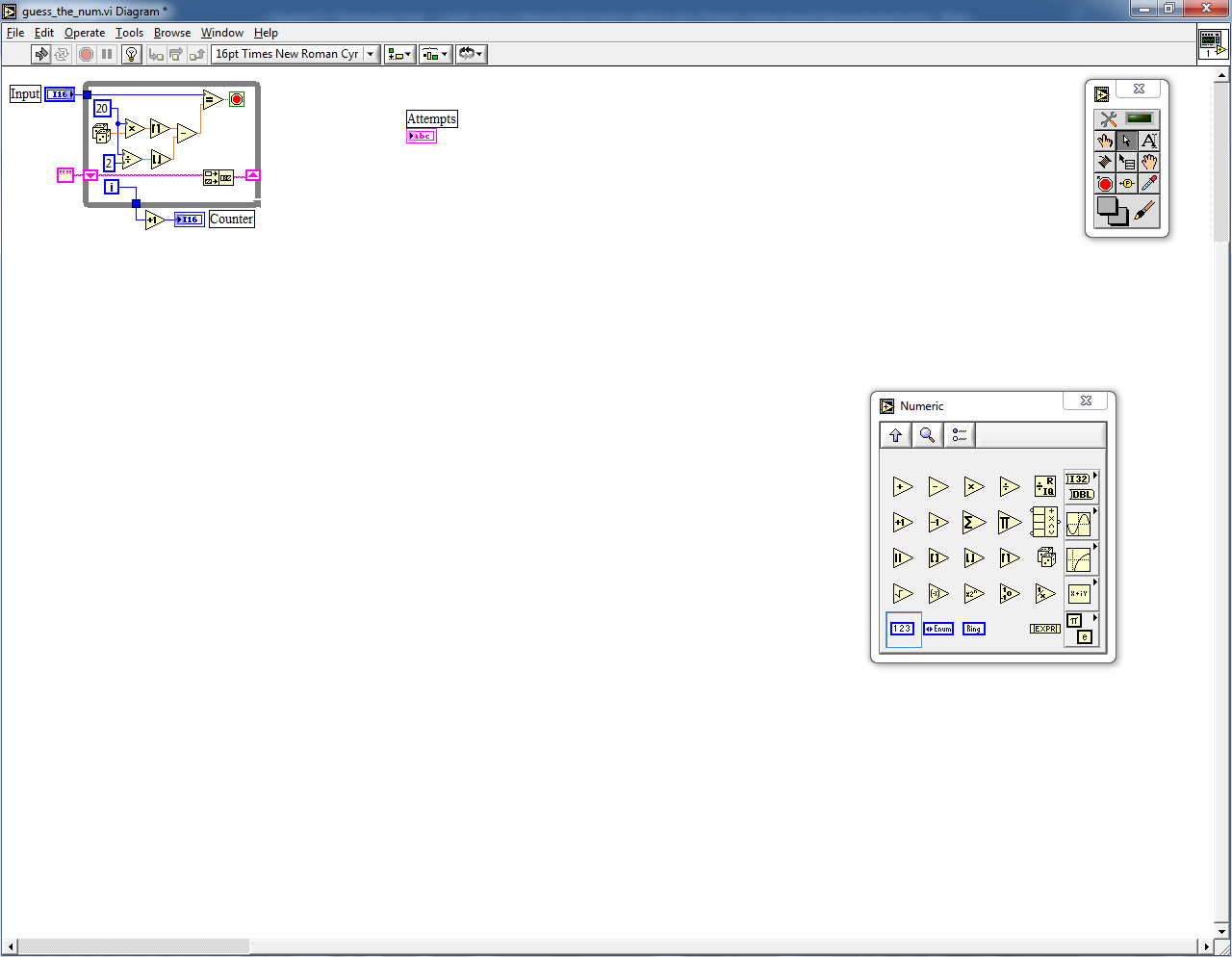


Рисунок 21 – Размещённый и подключённый по логике «к тому, что было записано ранее, дописываем…» элемент конкатенации строк на блок-диаграмме

Для формирования «чего-то нового», во-первых, потребуется разделитель, обеспечивающий удобство чтения элементов, во-вторых, строковое значение, полученное в результате конвертации / преобразования целого числа.

Итого, пойдём по порядку – добавим разделитель. В качестве разделителя принимается конструкция «запятая-пробел» («, »). Эта конструкция является строковой константой, потому на блок-диаграмме необходимо разместить именно строковую константу (Рисунок 22).

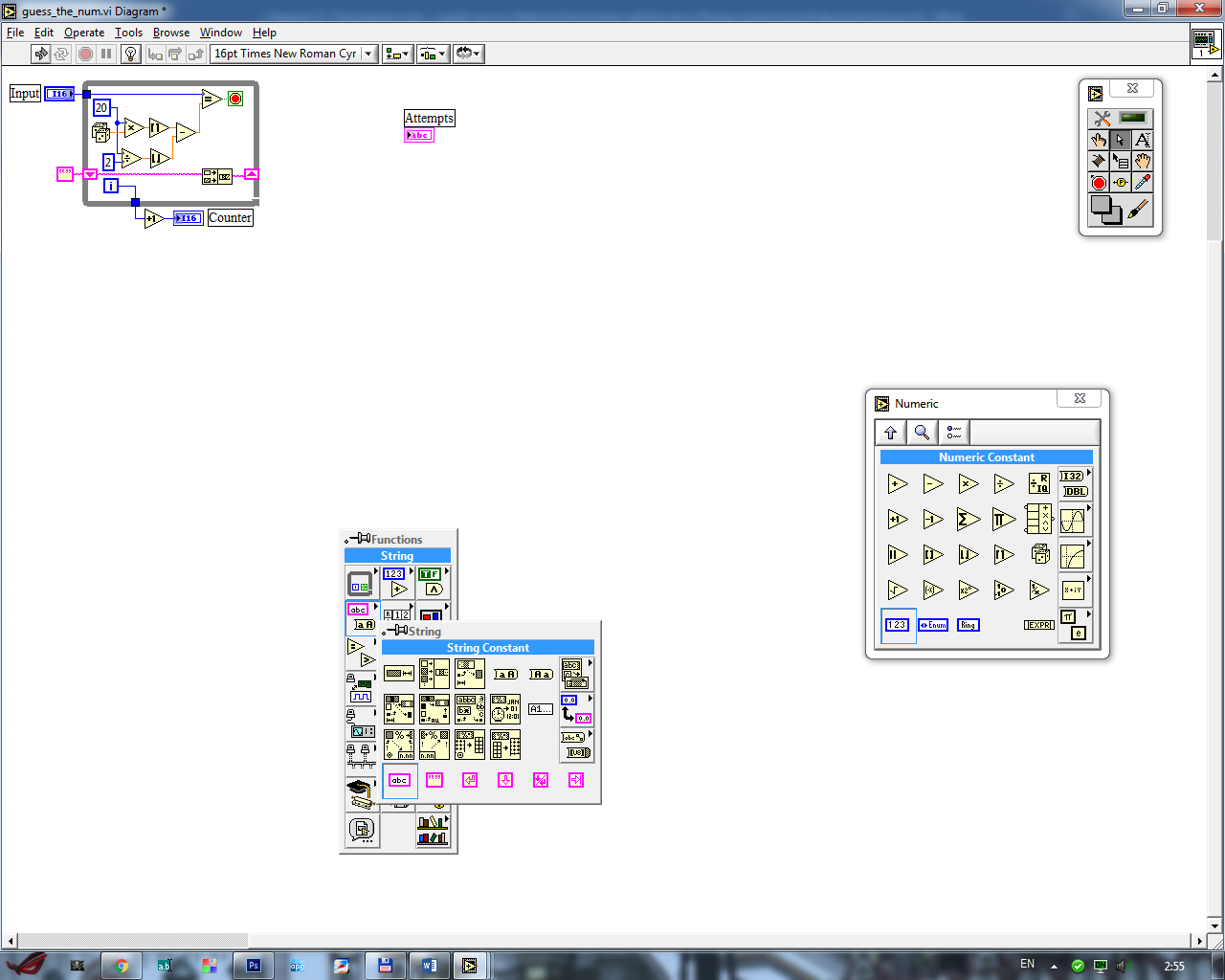


Рисунок 22 – Маркировка строковой константы (*String Constant*) раздела строк (*String*) функций, размещаемых на блок-диаграмме

Результат подключения разделителя представлен на Рисунке 23. Если запустить такую программу на исполнение, то на выходе, то есть на строковом индикаторе, можно наблюдать набор из запятых и пробелов в количестве попыток, предпринятых компьютером с целью угадать заданное пользователем число. Не хватает только численных значений, вставленных между ними.

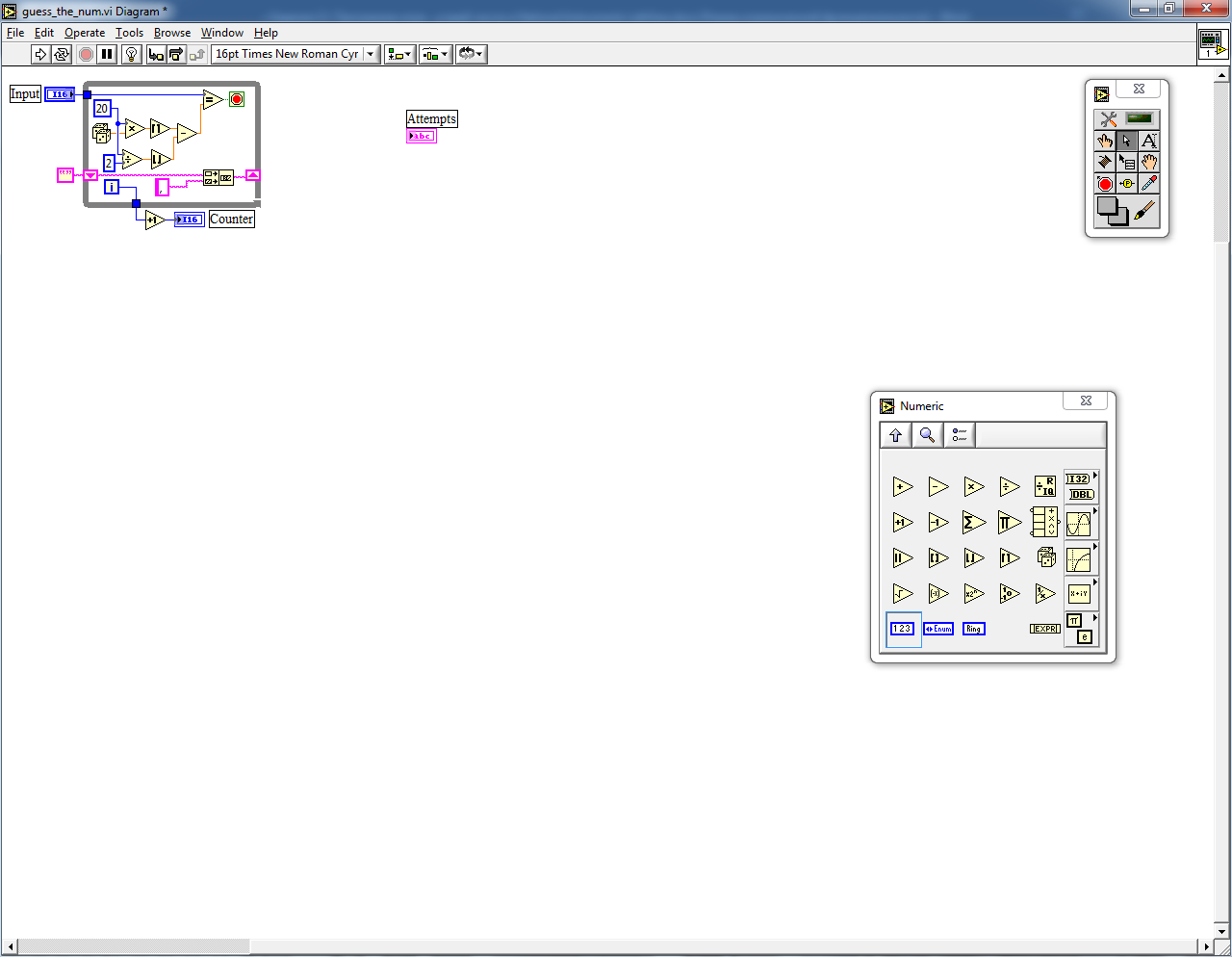


Рисунок 23 – Добавление строковой константы «, » к функции конкатенации

Для конвертации числа в строку используется один из специализированных конвертеров. Различные конвертеры из численного значения в строковое значение и обратно собраны в специальном подразделе строковых элементов (Рисунок 24). Необходим тот, который дословно называется «Число в десятичную строку». Подразумевается, что число на входе этой функции представлено значением в десятичной системе счисления и в том же виде, в десятичной же системе счисления, оно должно быть преобразовано в строку.

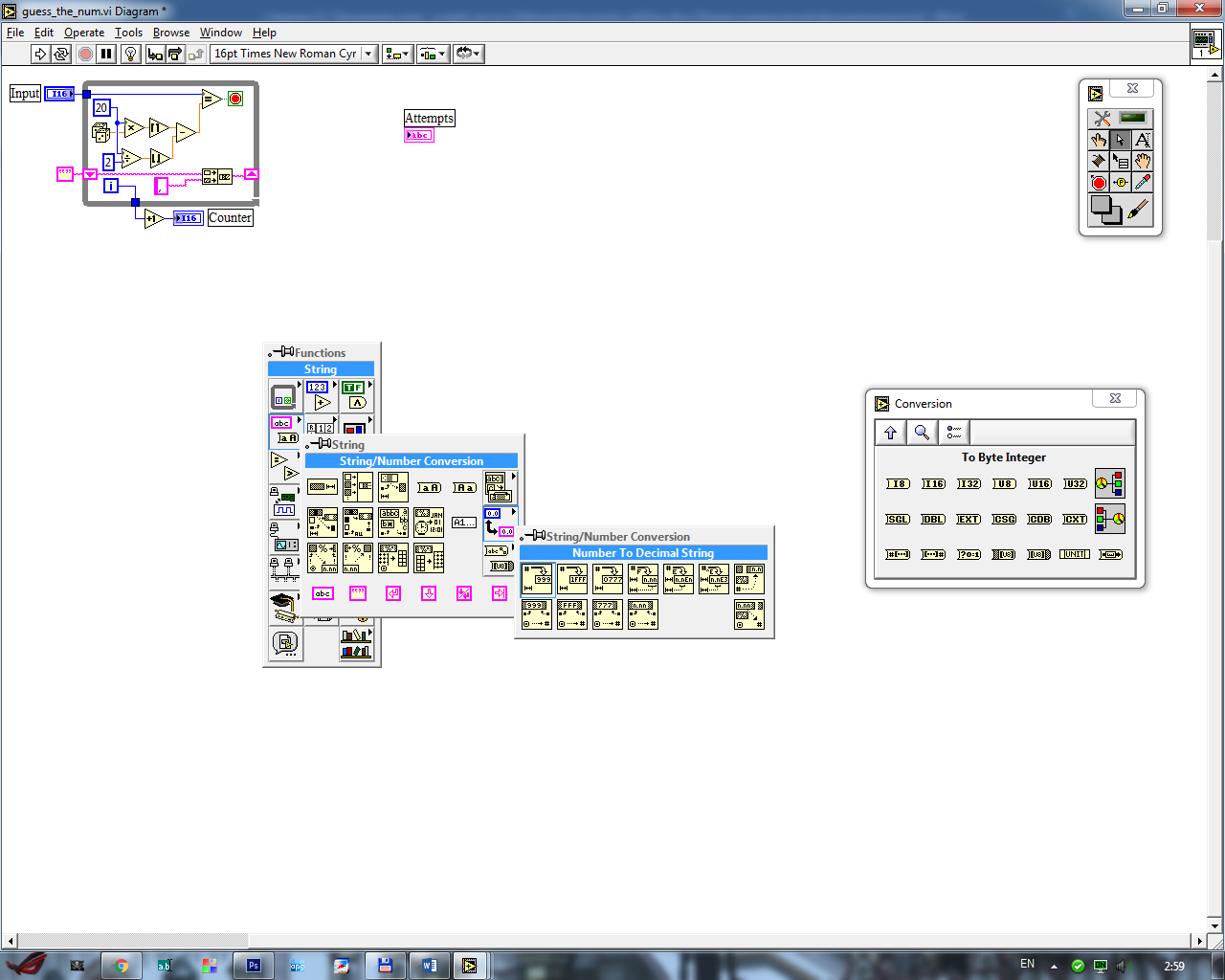


Рисунок 24 – Маркировка конвертера числового значения в строку (*Number To Decimal String*) подраздела строко-числовой конвертации (*String/Number Conversion*) раздела строк (*String*) функций, размещаемых на блок-диаграмме

С применением уже известного инструмента позиционирования / изменения размерности / выбора вытянем элемент конкатенации вниз и тем самым добавим ему ещё один вход. К этому входу и необходимо подключить сконвертированное в строку число (Рисунок 25).

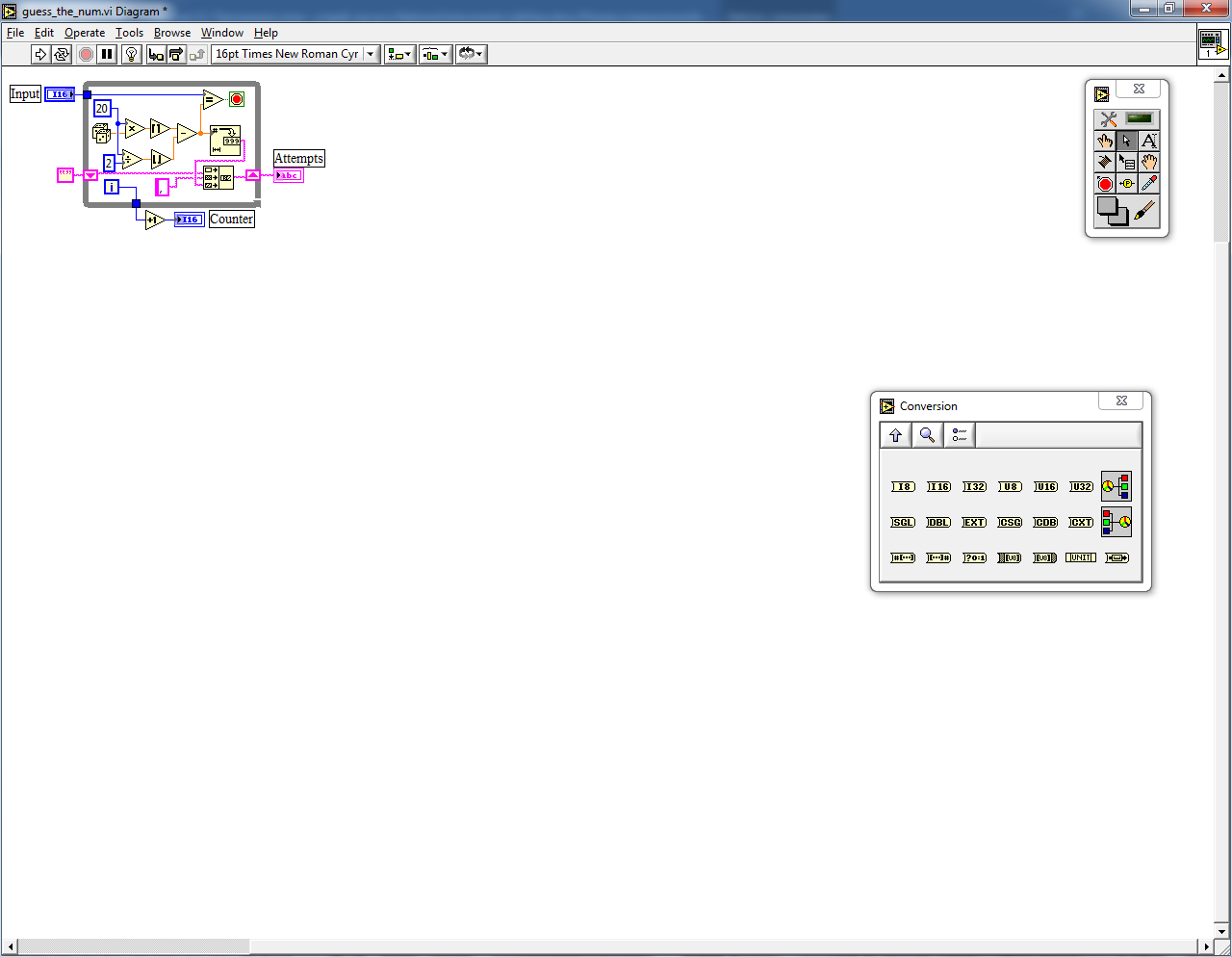


Рисунок 25 – Подключение преобразованного в строковый тип численного значения к функции конкатенации

В результате запуска можно увидеть некоторую неприятность (Рисунок 26). Действительно, на первой итерации мы к пустой строке дописываем запятую с пробелом и только потом интересующее нас численное значение. Первую запятую и пробел необходимо искоренить.

Есть два очевидных пути решения данной проблемы: либо организовать логику разветвления внутри цикла и на первой итерации записывать только сконвертированное в строку число, а на всех остальных итерация – разделитель и сконвертированное в строку число, либо вне цикла отсечь от результата первые два заведомо известных символа. Выберем для реализации второй подход.

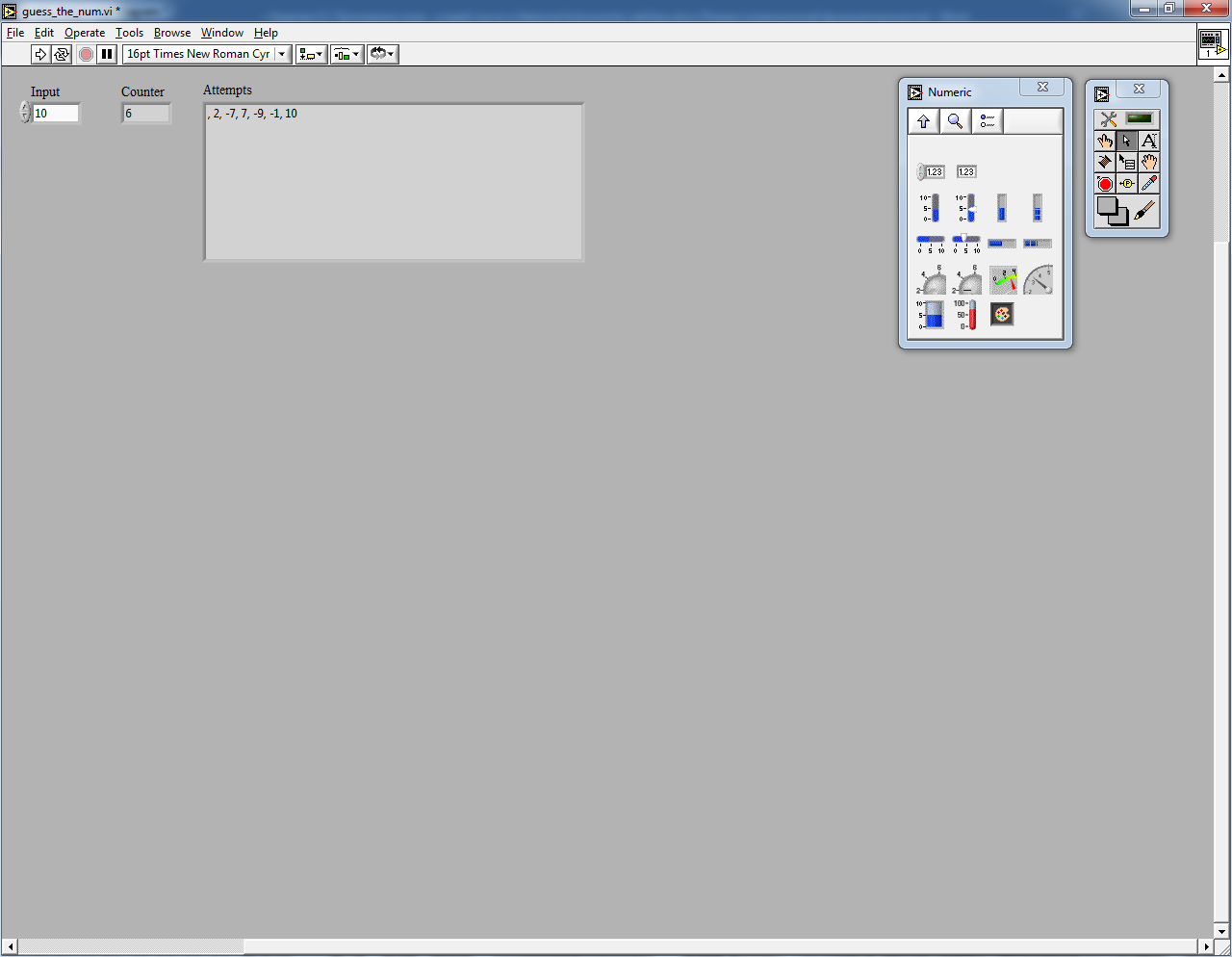


Рисунок 26 – Результат однократного запуска виртуального прибора

Для него потребуется функция вырезки подстроки из исходной строки (Рисунок 27). Эта функция принимает на вход саму строку, смещение от начала исходной строки (*offset*), длину вырезаемой строки (*length*). Входы сдвига и длины могут оставаться ни к чему не подключёнными. Если это так, то строка начинается с нулевого символа, а в качестве длины берутся все символы, которые остались после сдвига (*rest*). То есть без подключения к чему-либо численных входов строка на выходе функции та же, что и на входе.

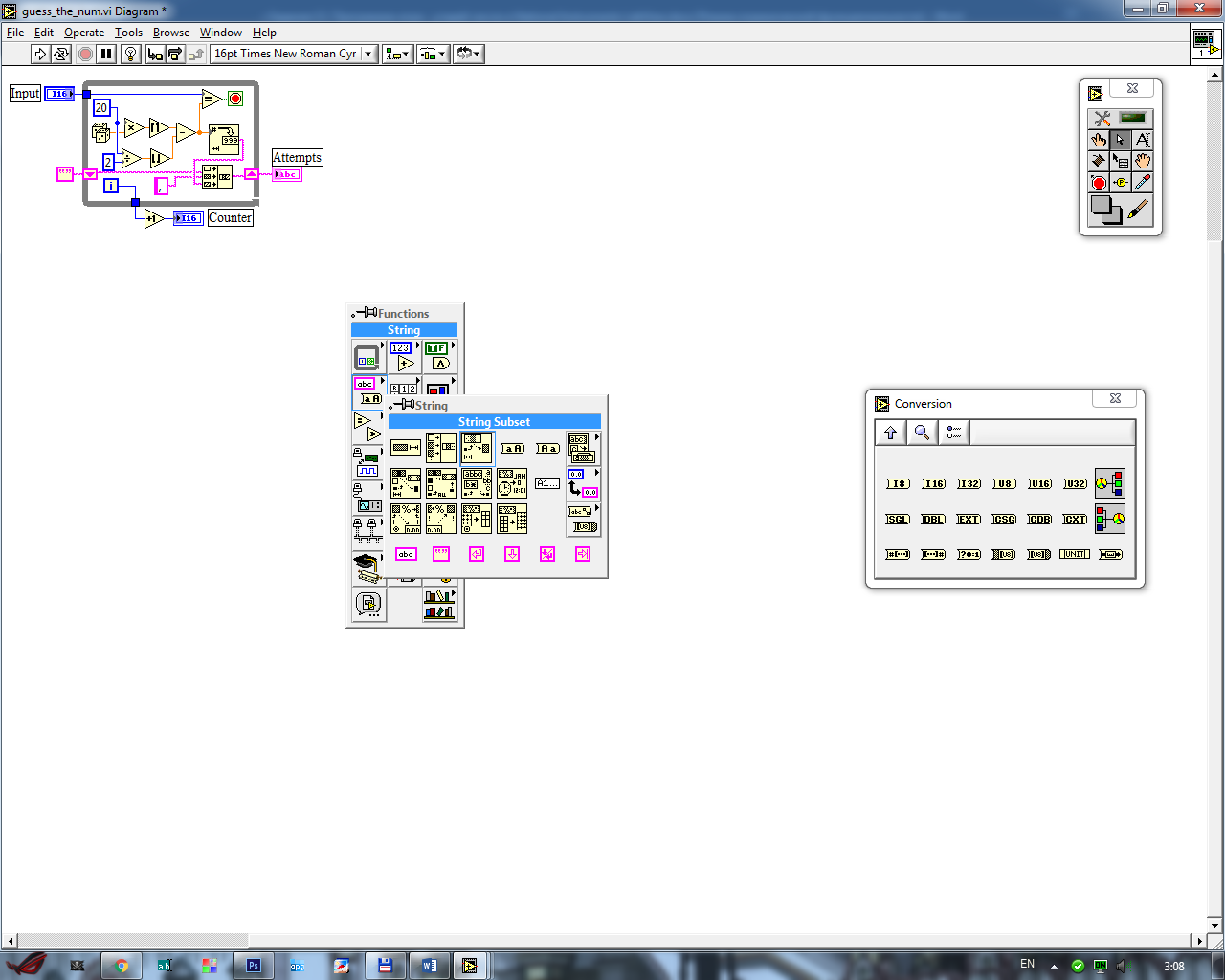


Рисунок 27 – Маркировка функции вырезки подстроки из строки (*String Subset*) раздела строк (*String*) функций, размещаемых на блок-диаграмме

Разместим вырезку подстроки на блок-диаграмме справа от циклической структуры и константой, равной двум, зададим сдвиг. Таким образом, выдадим функции вырезки подстроки команду начать подстроку со второго символа (учитывается обстоятельство, что счёт символов ведётся от нуля). На нулевой позиции – «,», на первой позиции – «пробел». Со второй позиции начинаются значимые для нас символы (Рисунок 28).

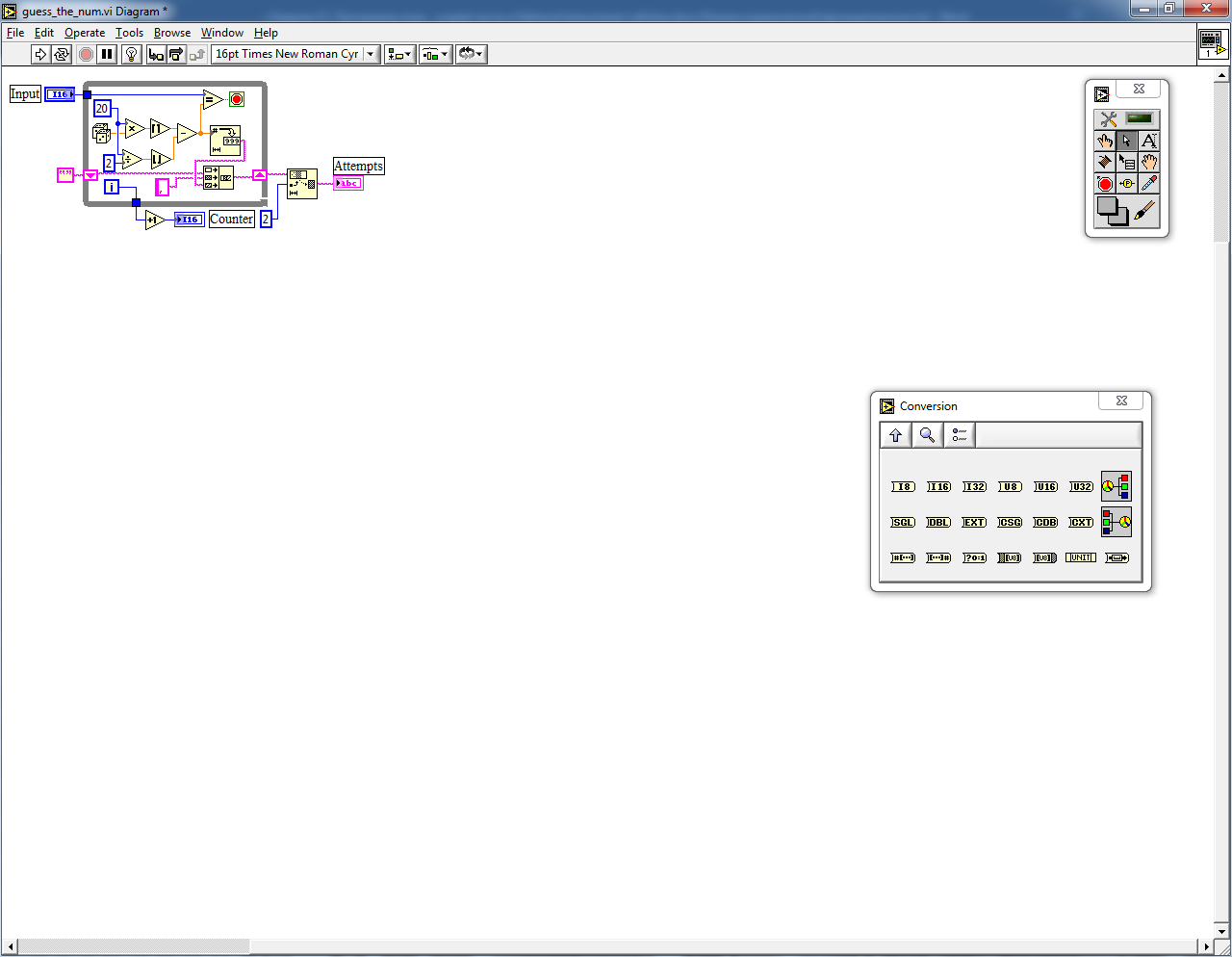


Рисунок 28 – Размещение функции вырезки подстроки из строки на блок-диаграмме

Полученный результат с учётом выполненных преобразований уже начинает отвечать предъявляемым к нему требованиям (Рисунок 29). Обучающимся же рекомендуется подумать над вопросом: включать или не включать угаданное значение в перечень? По логике вещей «предложение в качестве правильного ответа» – не есть сам правильный ответ, однако сам правильный ответ, все же, является гипотезой прежде, чем принимается решение о его корректности. В том случае, если обучающиеся принимают и обосновывают первый вариант философского размышления, то им необходимо отсечь от итоговой строки ещё и конструкцию после последней запятой включая последнюю запятую, либо если они принимают и обосновывают второй вариант философского размышления, то они оставляют кодовую конструкцию в представленном на Рисунке 28 виде.

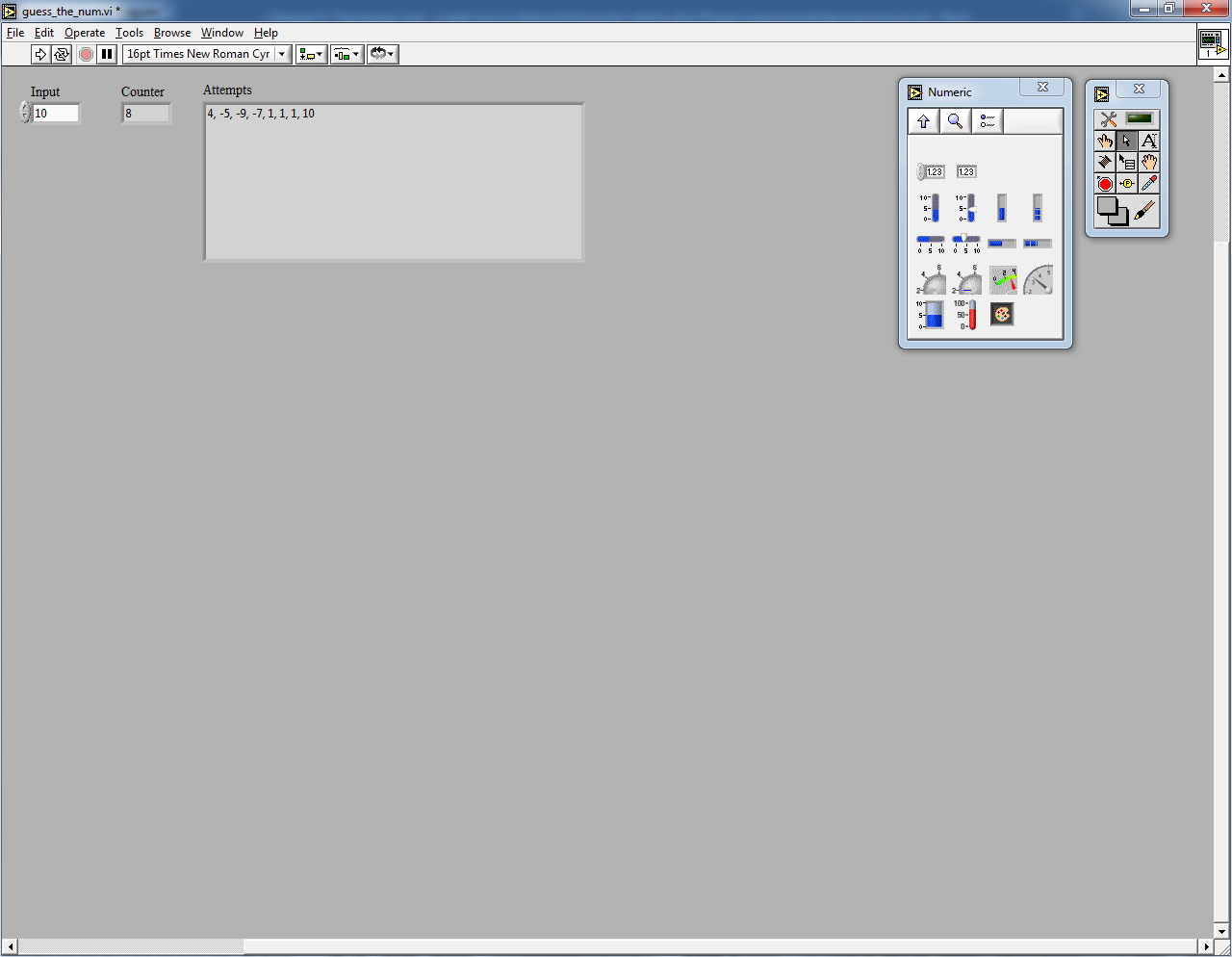


Рисунок 29 – Результат однократного запуска модифицированного виртуального прибора