ЛАБОРАТОРНАЯ РАБОТА №1. ЯЗЫК ST (STRUCTURED TEXT), РАБОТА С ТИПОМ ДАННЫХ EBOOL, INT, TIME И СЧЕТЧИКАМИ. УСТАНОВКА ПЕРЕДНЕГО ФРОНТА. ЗНАКОМСТВО С КОНСТРУКЦИЕЙ IF..THEN..ELSE..END_IF

Цель лабораторной работы:

1) Знакомство с различными конструкциями языка *ST*.
2) Решение задачи на включение и выключение индикатора на языке *ST*, работа с типом данных EBOOL.
3) Знакомство с функцией RE (установка восходящего фронта).
4) Команды SET и RESET.
5) Работа с различными видами счетчиков.
6) Команды INC и DEC.
7) Работа с типами данных INT и TIME. Системный бит %S6.

1.1 Основные теоретические сведения

Язык программирования ST - является одним из стандартных языков программирования МЭК 61131-3, который используется в программной среде Unity Pro. Данный язык является текстовым, работает с инструкциями, «выражениями». Под «выражениями» понимаются конструкции, состоящие из операторов и операндов, которые возвращают значения после выполнения.

Операторами являются специализированные символы, заложенные в программной среде для выполнения операций.

Операнды представляют собой переменные, литералы, входы/выходы функциональных блоков, адреса, многоэлементные переменные, эдементы многоэлементной переменной и т.д.

Код формируется в виде инструкций, которые используют для присваивания значений, возвращенных из выражений. В программной среде Unity Pro, рабочее окно редактора языка ST ограничено 300 символами, а размер секции не ограничен и зависит только от объема памяти программируемого логического контроллера. Рассмотрим основные конструкции языка ST.

1. Конструкция на условный оператор *if*.

```
IF  THEN
  ELSIF THEN ELSE
END_IF;
```

2. Конструкция с циклом *FOR*.

```
FOR TO BY  DO
END_FOR;
```

3. Конструкция с циклом WHILE.

```
WHILE DO
END_WHILE;
```

4. Конструкция с циклом REPEAT.

```
REPEAT
UNTIL
END_REPEAT;
```

5. Оператор множественного выбора CASE.

```
CASE  OF
  ELSE
END_CASE;
```

Далее рассмотрим примеры решения некоторых задач с помощью типовых конструкций языка ST.

1.2 Пример решения задач с помощью конструкции *if ... then ... else*

Задача №1.1 Необходимо включить индикатор, при нажатии кнопки. В решении задачи необходимо использовать функцию RE (установка переднего фронта).

Решение:

Для реализации данной программы выберем следующие переменные (таблица 1.1). Поскольку отладка программы будет осуществляться на симуляторе контроллера, физическая адресация переменных не задается.

Таблица 1.1 - Переменные для решения задачи (EDT)

№1	Название переменной (name)	Тип переменной (type)	Комментарий (comment)
1	button	ebool	кнопка нажатия лампочка
2	status_button	ebool	буферная переменная, необходимая для работы с функцией RE
3	light	ebool	индикатор

При работе с типом переменных *ebool*, для установки переднего фронта с помощью функции RE необходимо осуществить операцию перезаписи, чтобы в истории формата появилось значение 1. Данная процедура необходима только при работе с симулятором ПЛК.

Рассмотрим код программы:

```
status_button:=button; (*операция перезаписи для установки фронта*)
if RE(status_button) then light:=1; (*условие включения индикатора*)
   else light:=0; (*условие выключения индикатора*)
end_if;
```

На рисунке 1.1 представлена программа в среде Unity Pro.

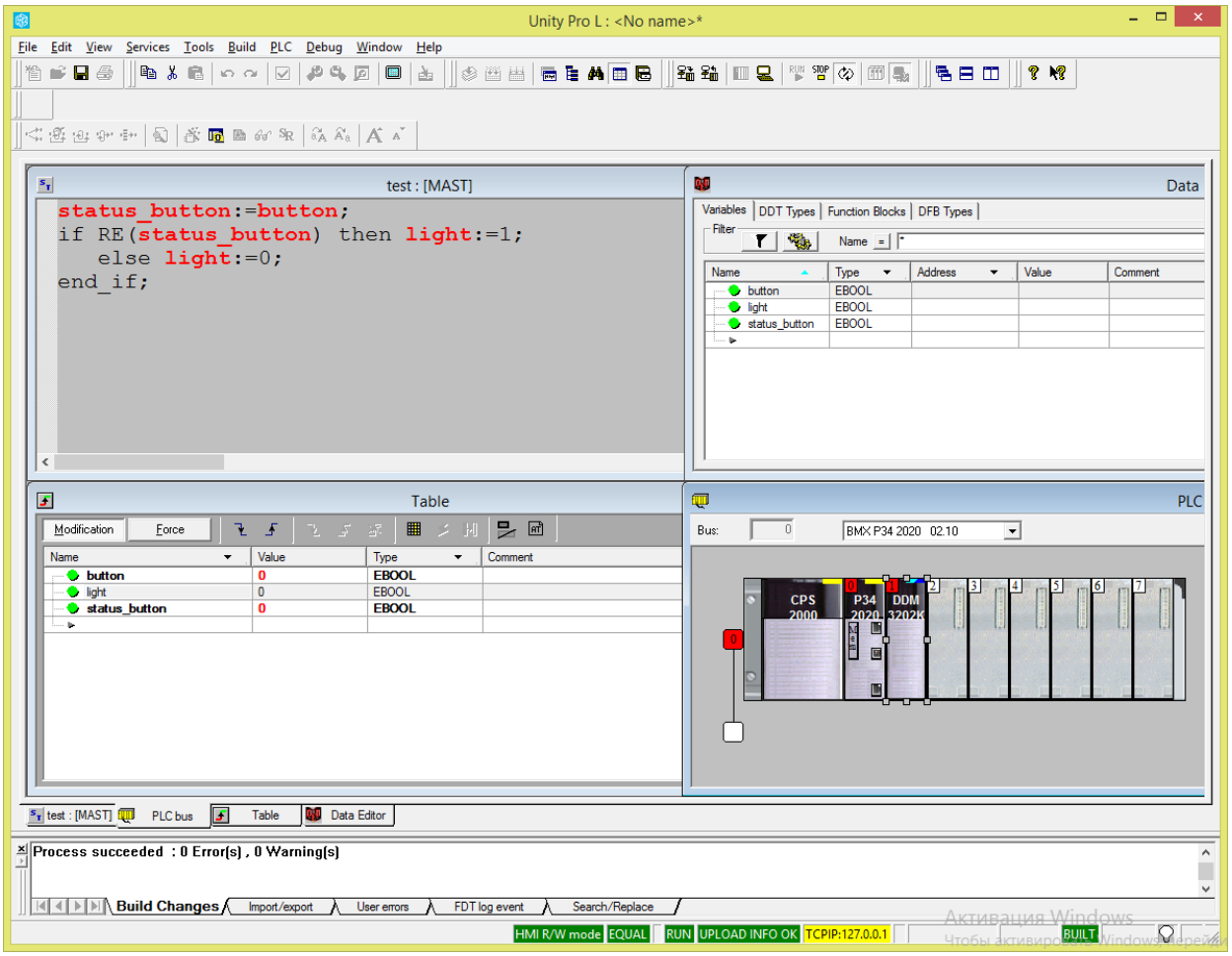

Рисунок 1.1 - Реализация программы в среде Unity Pro

Рассмотрим альтернативное решение задачи №1 с помощью операторов языка SET и RESET. Оператор SET устанавливает значение переменной на 1, оператор RESET устанавливает значение переменной на 0.

Таким образом код программы можно переписать следующим образом:

```
status_button:=button; (*операция перезаписи для установки фронта)
if RE(status_button) then SET(light); (*условие включения индикатора*)
   else RESET(light); (*условие выключения индикатора*)
end_if;
```

Возможно сокращение записи за счет замены выражения:

```
then light:=1; else  light:=0;
```

на выражение с помощью логического элемента NOT.

```
light:= not light;
```

Тогда код программы будет выглядеть так:

```
status_button:=button;
if RE(status_button) then light:= not light;
end_if;
```

Далее рассмотрим принцип работы со счетчиками на языке *ST*. В Unity Pro существует два типа счетчиков:

- подписанные (signed) - тип данных *INT*;
- неподписанные (unsigned) - тип данных *UDINT*;

Отличительной особенностью этих форматов является то, что подписанный счетчик типа *INT* (целочисленный формат) может принимать значение от 0 до 32768, а неподписанный счетчик от минус бесконечности до плюс бесконечности.

Рассмотрим пример инструкций по наращиванию значения переменной на 1:

```
counter_signed:= counter_signed+1; (*тип данных INT*)
counter_unsigned:= counter_unsigned+1; (*тип данных UDINT*)
```

Значение переменной можно увеличивать не только на 1, но и на другую величину. При уменьшении значения счетчика в выражении ставится знак «-». Для наращивания счетчика только на 1 в языке ST есть встроенные операторы INC и DEC. Пример инструкций с этими операторами приведен ниже:

```
INC (counter_signed) (*увеличивает значение счетчика на 1*)
DEC (counter_signed) (*уменьшает значение счетчика на 1*)
INC (counter_unsigned) (*увеличивает значение счетчика на 1*)
DEC (counter_unsigned) (*уменьшает значение счетчика на 1*)
```

Задача №1.2. Необходимо создать программу в которой при нажатии кнопки «плюс» будет наращиваться значение счетчика на 1, при нажатии на кнопку «минус», значение счетчика будет уменьшаться на 1.

Решение:

Для реализации данной программы выберем следующие переменные (таблица 1.2).

Таблица 1.2 - Переменные для решения задачи (EDT)

№1	Название переменной (name)	Тип переменной (type)	Комментарий (comment)
1	plus	ebool	кнопка при нажатии которой, значение счетчика будет наращиваться
2	minus	ebool	кнопка при нажатии которой, значение счетчика будет уменьшаться
3	status_plus	ebool	буферная переменная для работы с функцией RE для кнопки plus
4	status_minus	ebool	буферная переменная для работы с функцией RE для кнопки minus
5	counter	INT	счетчик

Рассмотрим код программы:

```
status_plus:= plus; (*операция перезаписи для установки фронта на кнопку plus*)
status_minus:=minus; (*операция перезаписи для установки фронта на кнопку minus*)
if RE(status_plus) then counter:=counter+1; (*инструкция для наращивания счетчика на 1*)
  if RE(status_minus) then counter:=counter-1; (*инструкция для уменьшения счетчика на 1*)
  end_if;
end_if;
```

Аналогичным образом можно переписать код программы через операторы INC и DEC.

```
status_plus:= plus;
status_minus:=minus;
if RE(status_plus) then INC(counter);
  if RE(status_minus) then DEC(counter);
  end_if; end_if;
```

На рисунке 1.2 представлено решение в среде Unity Pro.

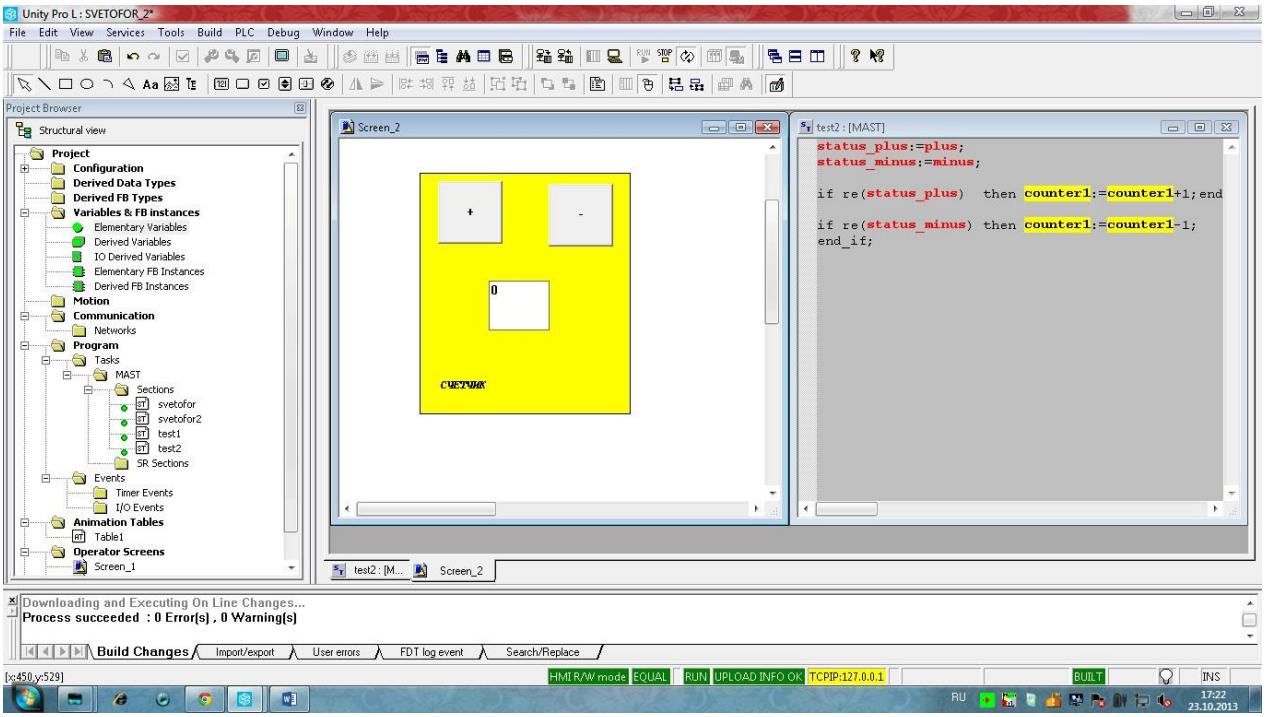

Рисунок 1.2 - Решение задачи №2 в среде Unity Pro

В программной среде Unity Pro для создания счетчика можно использовать системные биты. Рассмотрим системный бит %S6, который меняет интервал времени в 1 секунду.

Задача №1.3 Необходимо создать два новых счетчика, один счетчик будет иметь формат INT (целочисленный), второй TIME (время). Счетчики должны наращивать значение на 1 в промежутке времени равном 1 секунде с помощью системного бита %S6.

Решение:

Для реализации данной программы выберем следующие переменные (таблица 1.3).

Таблица 1.3 - Переменные для решения задачи (EDT)

№1	Название переменной (name)	Тип переменной (type)	Комментарий (comment)
1	counter_sec_int	INT	счетчик целочисленного формата
2	counter_sec_time	TIME	счетчик формата время
3	button	ebool	кнопка запуска работы счетчиков
4	status_button	ebool	буферная переменная для работы с функцией RE для кнопки button
5	sec1	ebool	переменная для хранения данных системного бита %S6

Рассмотрим код программы:

sec1:=**%s6**; (*интервал времени раз в 1 секунду*)
status_button:=button;

if RE(sec1) **then** counter_sec_int:=counter_sec_int+1; **end_if;**
if RE(sec1) **then** counter_sec_time:=counter_sec_time+t#1s; **end_if**;

Формат время в Unity Pro задается следующим образом: **t#1s**.
На рисунке 1.3 представлена реализация программы в среде Unity Pro.

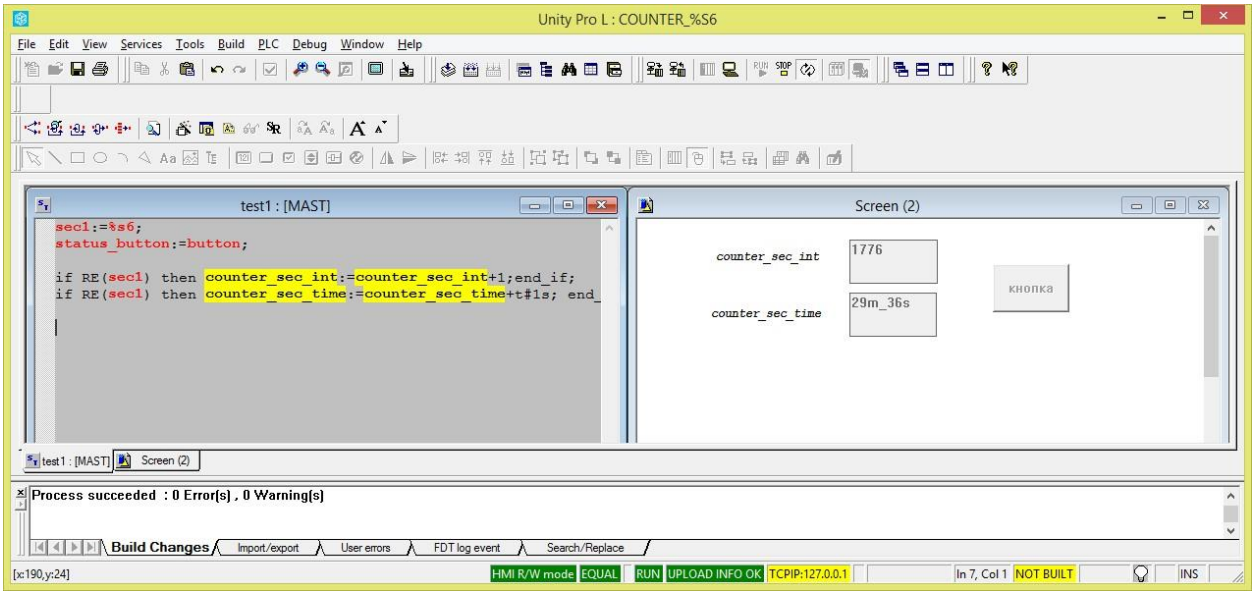

Рисунок 1.3 - Решение задачи №3 в среде Unity Pro

Разница между форматом INT и TIME очевидна. Формат время отчитывает минуты и секунды (Рисунок 1.3). Дополнительно для разработки программ можно использовать другие системные биты в среде Unity Pro (%S0 до %S124).

1.3 Порядок выполнения лабораторной работы №1

Задачи для самостоятельного выполнения:

Задача №1.4 Создайте программу в которой имеется два счетчика. Один счетчик целочисленного типа, второй формата время. Необходимо запрограммировать наращивание счетчиков на 60 для целочисленного и на 60 секунд для счетчика, работающего с форматом время. Наращивание счетчиков должно происходить в интервале времени 100 ms. Запуск работы счетчиков будет осуществляться после нажатия кнопки "СТАРТ", остановка работы счетчиков после нажатия кнопки "СТОП". Сделайте визуализацию программы.

Задача №1.5 Даны два индикатора различных цветов. После нажатия кнопки "СТАРТ" загорается первый индикатор, спустя 5 секунд включается второй индикатор. Запрограммируйте данную последовательность с помощью счетчика. Сделайте визуализацию в экране реального времени.

Задача №1.6 Дан счетчик, значение которого наращивается на 2 с интервалом в 1 секунду, как только достигается значение 100, счетчик останавливается. Запуск работы счетчика осуществляется с помощью кнопки "СТАРТ". Сделайте визуализацию работы программы.

Задача №1.7 По умолчанию значение переменной является 100. После нажатия кнопки "СТАРТ" начинается обратный отчет счетчика, как только значение переменной достигает 5, счетчик останавливается и загорается зеленый индикатор. Сделайте визуализацию работы программы.

1.4 Оформление отчета по результатам выполненных работ.

Отчет должен включать:
- Конфигурацию контроллера;
- Таблицу переменных программы;
- Алгоритм выполнения работ;
- Выводы по результатам практикума.
- Решение задачи в пакете Unity Pro в формате .STU.

1.5 Контрольные вопросы

1) Опишите основные конструкции языка ST.
2) Для чего нужна команда RE?
3) Что такое системный бит?
4) Опишите назначение системного бита %S6.
5) Какие бывают счетчики?
6) Как задается время в среде Unity Pro?
7) Для чего нужны команды SET и RESET?
8) Какую функцию выполняют команды INC и DEC?