**4 РАЗРАБОТКА ПРОГРАММНОЙ ЧАСТИ**

**АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ**

**4.1 Разработка программы ПЛК**

Программа для ПЛК *SIMATIC S*7-1200 разрабатывается в интегрированной среде разработки *TIA Portal* 15 от компании *Siemens AG*. В *TIA Portal* 15 интегрированы следующие компоненты: *Simatic Step* 7 (инструмент для программирования контроллеров), *WinCC* (инструмент для создания *HMI*), *PLCSIM* (симулятор работы ПЛК). Разрабатываемый проект состоит из программы ПЛК и проекта для графической панели оператора (*HMI*).

Стартовое окно *TIA Portal* 15 обеспечивает доступ к основным элементам проекта. «Проект» представляет собой структуру, которая содержат все программы и данные, необходимые для выполнения задачи автоматизации. Через главное окно можно открыть существующий проект, создать новый проект или произвести преобразование проекта.

В разделе *Devices & network* производится конфигурирование контроллеров, зада­ются модули, н производится их настройка.

В разделе *PLC programming* разрабатывается управляющая программа, формируется структура программы, создаются программные блоки.

В разделе *Visualization* собраны инструменты по разработке человеко-машинных ин­терфейсов (*HMI*) с использованием *Simatic WinCC*.

В разделе *Online & Diagnostics* возможно подключение к программируемому кон­троллеру. управление режимами его работы *CPU*. тестирование работы управляющей про­граммы.

Режим *Project view* позволяет отобразить проект в виде иерархической структуры, причем содержимое и вид окна рабочей области может меняться в зависимости от выпол­няемых действий. Окно содержит рабочую область, в которой производится конфигурирование оборудо­вания контроллера, каталог модулей, отображение свойств устанавливаемых модулей.

Если в структуре проекта будет выбран программный блок, то в окне разработки будет показана программа, а вспомогательные окна будут содержать элементы для формирования программы. Такой подход облегчает переключение окон при разработке проекта. Разработчику всегда доступно дерево проекта, а при редактировании необходимого элемента, дополнительные окна с инструментами редактирования появляются автоматически.

Для создания нового проекта *TIA Portal* 15 необходимо выбрать действие *Create new proje*ct (создать новый проект), после чего указать название проекта, задать каталог на диске, в котором будут храниться файлы проекта, и нажать на экране кнопку *Create*.

Следующий шаг ‒ конфигурация используемого аппаратного оборудования ‒ *Devices & Networks/Configure a device*. Необходимо добавить в проект (*Add new device*) модуль процессора *CPU* 1215*С* *DC/ DC/DC* с номером 6*ES*7 215-1*AG*40-0*XB*0. При этом в поле *Device name* оставляем автоматически предложенное имя (PLC\_1) (рисунок 4.1).

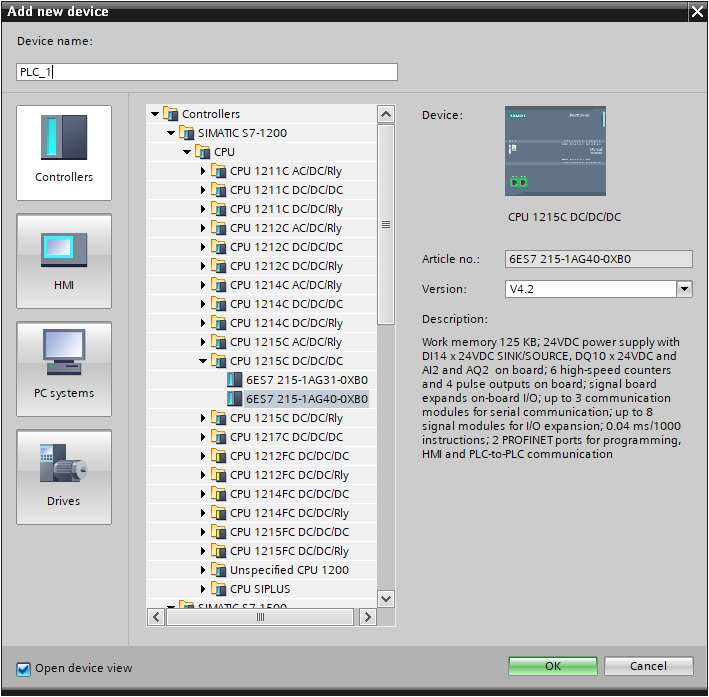


Рисунок 4.1 – Выбор модуля процессора *CPU* 1215*С*

После добавления *CPU* 1215*С* представление проекта автоматически изменится на проектно-ориентированное, в котором в рабочей области окна появится графическое изображение добавленного в проект контроллера. Щелкнув на нем правой кнопкой мыши и выбрав во всплывающем меню пункт *Properties* (свойства), в нижней части экрана в окне инспектора свойств объекта можно просматривать и нужным образом настраивать свойства *CPU*. *CPU* 1215*С* снабжен двумя портами *PROFINET* для обмена данными через сеть *PROFENET*. Для обмена данными через сети *RS*-485 имеется коммуникационный модуль, встраиваемый в *CPU*.

В окне свойств устанавливаются следующие параметры:

* интерфейсы *PROFINET*: установка IP-адресов (PN1: 192.168.0.1; PN2: 192.168.0.2).
* *DI*, *DO*, и *AI*: настройка поведения встроенных цифровых и аналоговых входов и выходов;
* время цикла 100 мс.

Далее добавляются из библиотеки модуль ввода аналоговых сигналов *SM*1231 (6ES7 231-4HF32-0XB0). После этого окно конфигурации примет окончательный вид (рисунок 4.2):

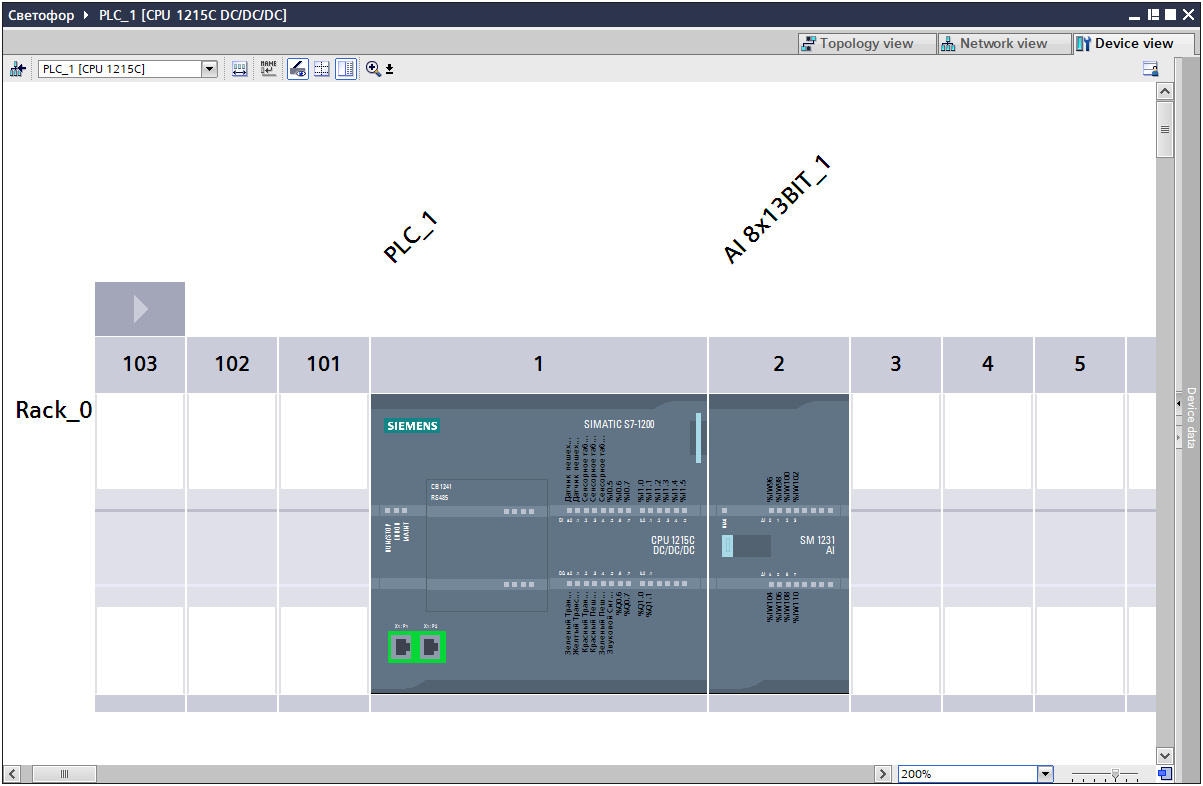


Рисунок 4.2 – Конфигурация контроллера *SIMATIC S*7-1200 для проекта автоматизации в *TIA Portal* 15

После настроек конфигурации проекта реализуется программный код проекта для ПЛК.

Программа ПЛК по структуре состоит из нескольких блоков. Исполнение программы пользователя *CPU* поддерживает следующие виды блоков, позволяющие создать эффективную структуру пользовательской программы:

* Организационные блоки (*ОВ*) определяют структуру программы. Некоторые *ОВ* имеют предопределенное поведение и стартовые события, но вы можете также создавать *ОВ* со своими собственными стартовыми событиями.
* Функции (*FC*) и функциональные блоки (*FB*) содержат программный код, соответствующий конкретным задачам или комбинациям параметров. Каждая функция и каждый функциональный блок предоставляет в распоряжение набор входных и выходных параметров для совместного использования данных с вызываемым блоком. *FB* использует также связанный с ним блок данных (называемый экземплярным *DB*) для сохранения данных о состоянии во время исполнения, которые могут быть использованы другими блоками в программе.
* Блоки данных (*DB*) хранят данные, которые могут быть использованы программными блоками.

Исполнение программы пользователя начинается одним или несколькими необязательными организационными блоками (*ОВ*), которые после перехода в режим *RUN* обрабатываются один раз, затем следует один или более ОВ программного цикла, которые обрабатываются циклически. ОВ может быть также поставлен в соответствие прерывающем событию, которое может быть стандартным событием или событием-ошибкой; затем он исполняется, когда происходит соответствующее событие.

Функция (*FC*) или функциональный блок (*FB*) − это блок с кодом программы, который может быть вызван из ОВ или из другой функции или другого функционального блока. При этом возможны следующие уровни вложения:

* 16 из циклического *ОВ* или *ОВ* запуска;
* 4 из *ОВ* прерываний с задержкой, *ОВ* циклических прерываний, *ОВ* аппаратных прерываний, *ОВ* ошибок по времени или *ОВ* диагностируемых ошибок *FC* не ставятся в соответствие никакому конкретному блоку данных (*DB*), тогда как *FB* непосредственно связаны с *DB* и используют этот *DB* для передачи параметров и сохранения промежуточных значений и результатов.

Размер пользовательской программы, данных и конфигурации ограничен имеющейся в распоряжении загрузочной памятью и рабочей памятью в *CPU*. В рамках свободной рабочей памяти число поддерживаемых блоков не ограничено. Каждый цикл включает в себя запись выходов, чтение входов, исполнение команд программы пользователя и выполнение обслуживания системы или фоновая обработка. Этот цикл называется также циклом сканирования или просто сканированием. Сигнальная плата, сигнальные и коммуникационные модули обнаруживаются и регистрируются только при запуске.

По структуре программа построена из организационного блока *ОВ*1, блока циклических прерываний *OB*30 и стартового блока OB100. Главная программа располагается в организационном блоке *ОВ*1, который всегда обрабатывается центральным процессором. Начало пользовательской программы идентично первому сегменту (сети, *network*) в *ОВ*1. По завершению обработки *ОВ*1 (конец программы) *CPU* передает управление операционной системе, и после вызова различных функций операционной системы, таких как обновление образа процесса, центральный процессор снова вызывает *ОВ*1.

Событиями, которые могут вмешиваться в работу программы, являются прерывания *(interrupts*) и ошибки (*errors*). В блоке циклических прерываний *OB*30 с интервалом 10 мс, обрабатываются сигналы с устройств, подключенных к контроллеру через RS-485. Для установки соединения с датчиком Аркен-Кросс вызывается блок MB\_COMM\_LOAD (см. рисунок 4.3).

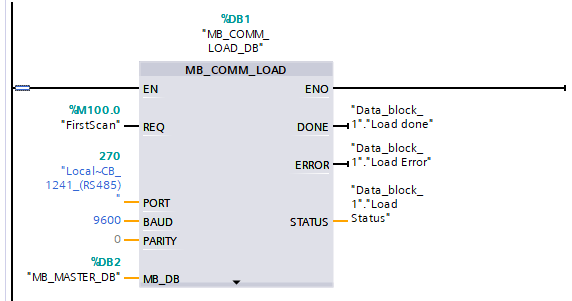


Рисунок 4.3 – Блок MB\_COMM\_LOAD

При первом включении блока MB\_COMM\_LOAD на вход REQ приходит положительный фронт, что заставляет этот блок создать подключение к датчику Аркен-Кросс с заданными параметрами.

Как только соединение с датчиком установлено, вызывается следующий блок MB\_MASTER (см. рисунок 4.4). Этот блок один раз в 10 мс считывает биты данных, посылаемых датчиком, и сохраняет их в блок данных (ArcenCrossData).

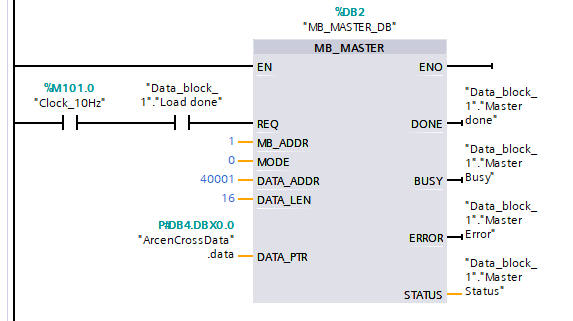


Рисунок 4.4 – Блок MB\_MASTER

Далее следует операция обнаружения транспорта. (см. рисунок 4.5). Когда датчик Аркен-Кросс определяет транспорт, логическая единица заносится в один из битов данных, тогда значение битов данных (ArcenCrossData.data[1], ArcenCrossData.data[1]) будет больше нуля и биту памяти (Транспортный датчик 1) приписывается логическая единица.



Рисунок 4.5 – Операция обнаружения транспорта

В блоке *ОВ*1 реализуется вся логика. Согласно разработанному алгоритму, обрабатываются состояния дискретных входов ПЛК и переменных. Программа реализована на языке *LAD*.

После загрузки ПЛК в программе вызывается блок *OB100*. При включении контроллера номер шага по умолчанию должен быть равен единице и количество проверок должно быть равно нулю. Таким образом осуществляется установка значений переменных (ШАГ, n) по умолчанию, значения которых описаны в *Network* 1 (см. рисунок 4.6).

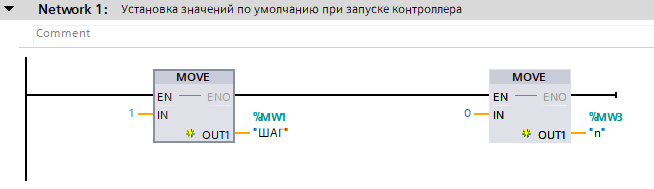


Рисунок 4.6 – Установка значений по умолчанию

После выполнения блока OB100 в программе циклически вызывается блок *OB*30 и в блоке *ОВ*1 реализуется управление работой пешеходного перехода. При запуске блока *OB1* переменная (ШАГ) равняется единице, производиться проверка транспортных светофоров (см. рисунок 4.7).

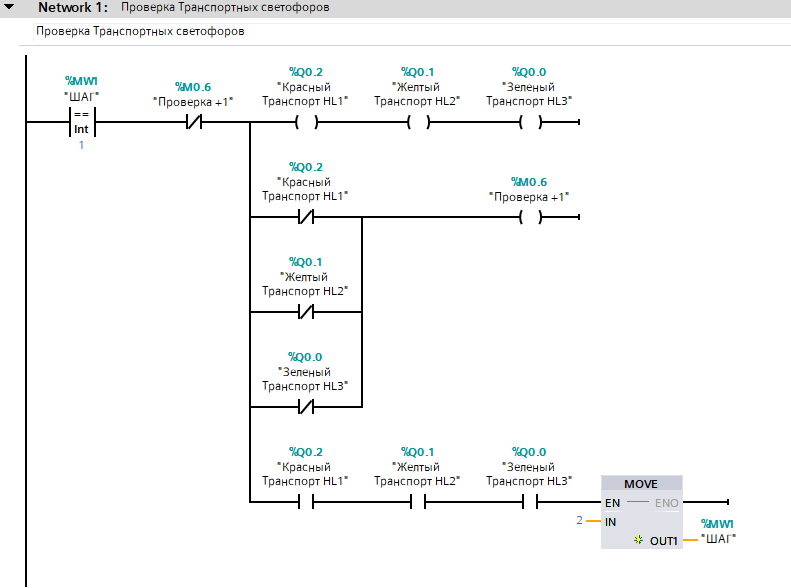


Рисунок 4.7 – Проверка транспортных светофоров (*Network 1)*

Посылается сигнал включения всех транспортных светофоров, если все транспортные светофоры включились успешно, то программа присваивает переменной (ШАГ) значение двойки и переходит на следующий шаг. Если хоть один транспортный светофор не включился, биту (Проверка +1) присваивается логическая единица, нормально замкнутый контакт разрывает цепь, биту (Проверка +1) присваивается логический ноль и проверка начинается заново.

Когда переменная (ШАГ) равняется двойке, производиться проверка пешеходных светофоров (см. рисунок 4.8).

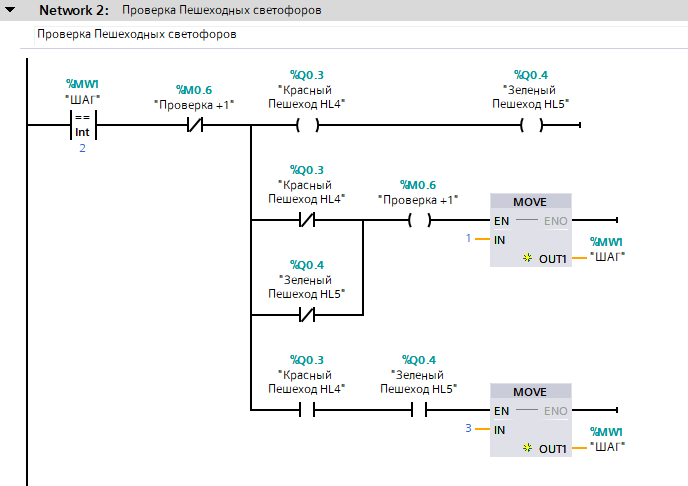


Рисунок 4.8 – Проверка пешеходных светофоров (*Network 2)*

Посылается сигнал включения всех пешеходных светофоров, если все пешеходные светофоры включились успешно, то программа присваивает переменной (ШАГ) значение тройки и переходит на следующий шаг. Если хоть один пешеходный светофор не включился, биту (Проверка +1) присваивается логическая единица и цикл проверки переходит на (ШАГ) номер один, проверка начинается заново.

Когда переменная (ШАГ) равняется тройке, производиться проверка звуковых сигнализаторов (см. рисунок 4.9).

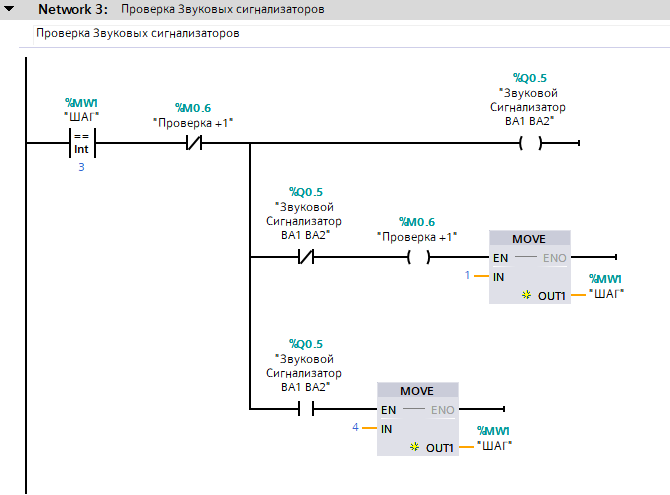


Рисунок 4.9 – Проверка звуковых сигнализаторов (*Network 3)*

Посылается сигнал включения всех звуковых сигнализаторов, если все звуковые сигнализаторы включились успешно, то программа присваивает переменной (ШАГ) значение четверки и переходит на следующий шаг. Если звуковой сигнализатор не включился, биту (Проверка +1) присваивается логическая единица и цикл проверки переходит на (ШАГ) номер один, проверка начинается заново.

Подсчет проверок реализован на счетчике CTU (см. рисунок 4.10).

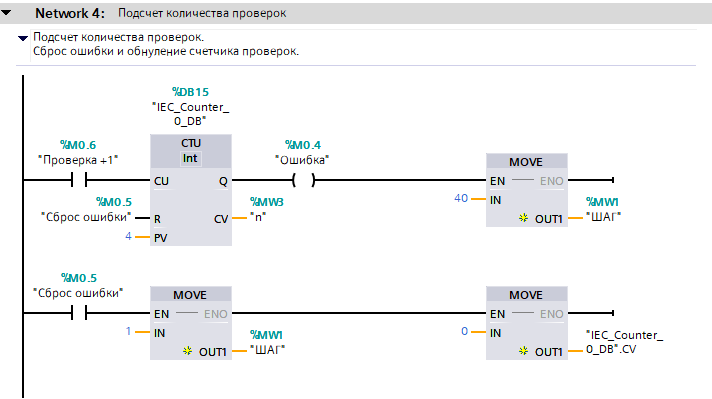


Рисунок 4.10 – Подсчет количества проверок (*Network 4)*

Если при включении ряда устройств, хотя бы одно из них не будет включено, в переменную (Проверка +1) записывается логическая единица, что вызывает повышение текущего значения счетчика на единицу. При превышении количества допустимых проверок в переменную (Ошибка) заноситься логическая единица и в переменную (ШАГ) заносится значение сорок и включается режим желтого мигания (см. рисунок 4.11).

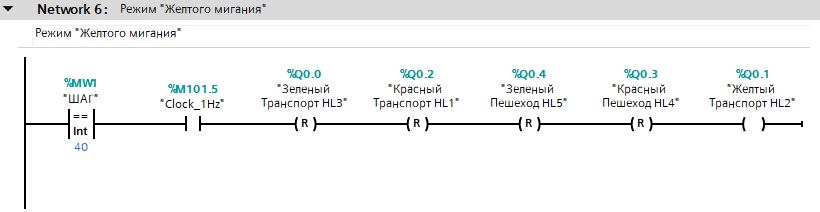


Рисунок 4.11 – Режим "Желтого мигания" (*Network 6)*

В режиме "Желтого мигания" раз в секунду переменно включаются/выключаются желтые транспортные сигнализаторы.

Также в *Network 5* создается отчёт об ошибке, который отправляется на АРМ оператора в виде Excel таблицы (см. рисунок 4.12).

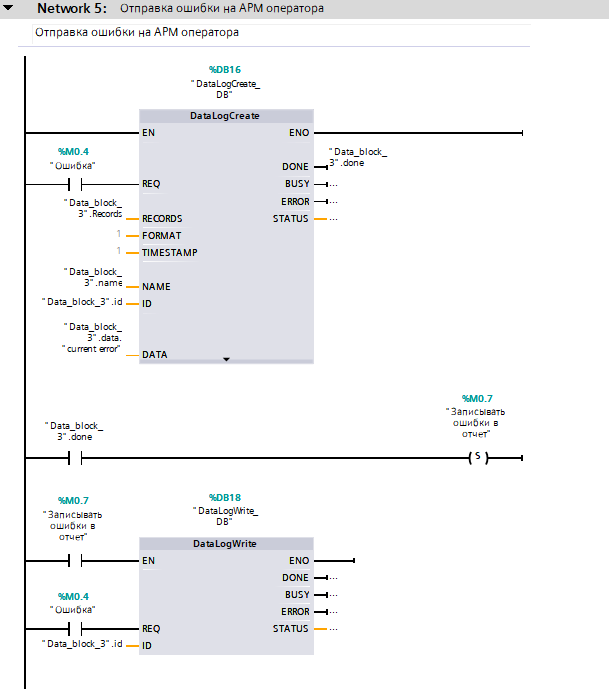


Рисунок 4.12 – Отправка ошибки на АРМ оператора (*Network 5)*

Когда на вход (REQ) блока DataLogCreate поступает положительный фронт, он создает лог файл, при успешном создании лог файла на выходе (DONE) будет положительный фронт, который записывается в (Data\_block\_3.done). Положительный фронт бита (Data\_block\_3.done) активирует бит (Записывать ошибки в отчёт), который в свою очередь положительным фронтом активирует блок DataLogWrite. Когда на вход (REQ) DataLogWrite поступает положительный фронт, он заносит оповещение о наличии ошибки в лог файл.

Когда переменная (ШАГ) равняется четверке программа пешеходного перехода разрешает движение транспорту, запрещает движение пешеходам, проверяет условия наличия пешехода в зоне действия датчика присутствия (SQ1, SQ2) и срабатывание вызывного табло (SQ3, SQ4, SQ5) (см. рисунок 4.13).

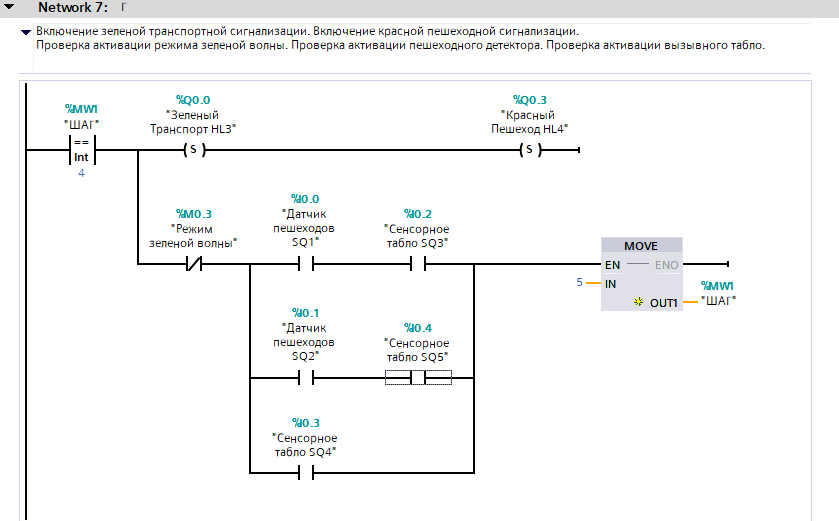


Рисунок 4.13 – запуск пешеходного перехода (*Network 7)*

При появлении пешехода и срабатывании вызывного табло программа присваивает переменной (ШАГ) значение пятерки и переходит на следующий шаг. При активации оператором режима зеленой волны, сигналы, поступающие с датчиков, не влияют на программу.

Когда переменная (ШАГ) равняется пятерке запускается переменное включение/выключение зеленой транспортной сигнализации в течении пяти секунд с интервалом в одну секунду (см. рисунок 4.14).

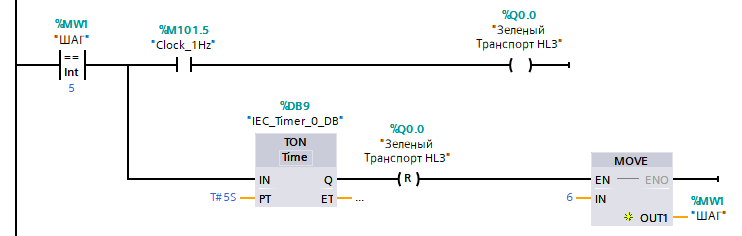


Рисунок 4.14 – Мигание зеленой транспортной сигнализации (*Network 8)*

Через пять секунд переменной (ШАГ) присваивается значение шесть, и программа переходит на следующий шаг.

Когда переменная (ШАГ) равняется шести включается желтая транспортная сигнализация и по окончании трех секунд выключается желтая и включается красная транспортная сигнализация, транспорту движение запрещено (см. рисунок 4.15).

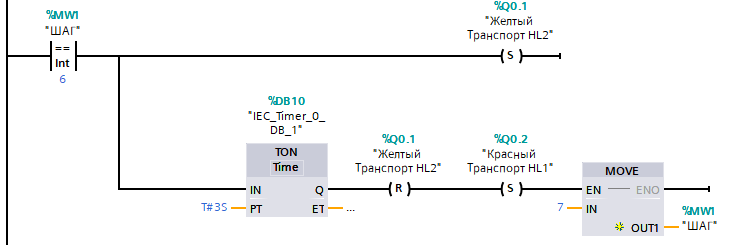


Рисунок 4.15 – Включение желтой и красной транспортных сигнализаций (*Network 8)*

Переменной (ШАГ) присваивается значение семь, и программа переходит на следующий шаг.

Когда переменная (ШАГ) равняется семи, через 2 секунды после включения шага включается зеленая пешеходная сигнализация вместо красной и раз в одну секунду работают звуковые сигнализаторы, пешеходам движение разрешено (см. рисунок 4.16).

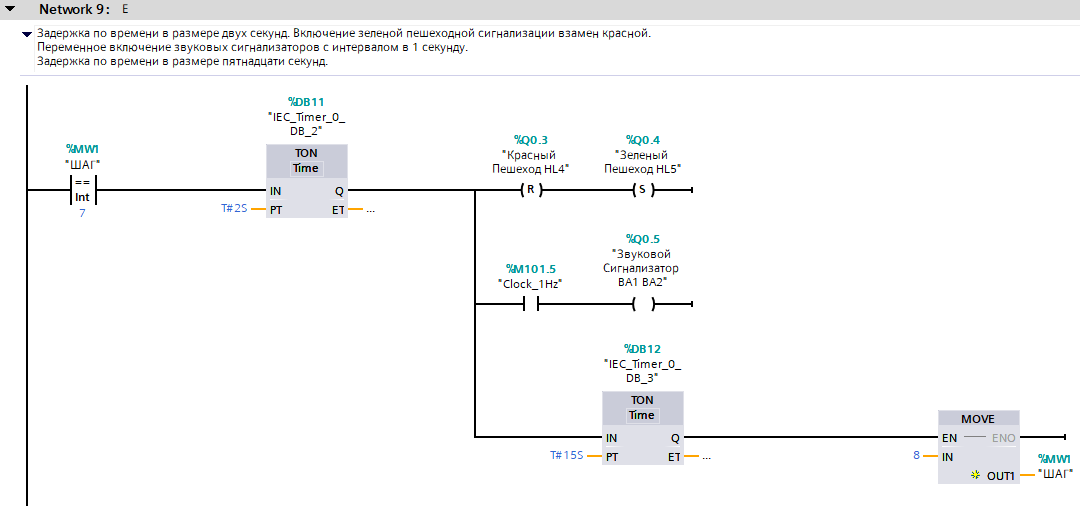


Рисунок 4.16 – Включение зеленой пешеходной сигнализации и звукового сигнализатора (*Network 9)*

По прошествии пятнадцати секунд переменной (ШАГ) присваивается значение восемь, и программа переходит на следующий шаг.

Когда переменная (ШАГ) равняется восьми, зеленая пешеходная сигнализация включается/выключается раз в секунду в течении пяти секунд. Звуковая сигнализация переменно включается/выключается два раза в секунду в течении 5 секунд. По прошествии пяти секунд выключается зеленая и включается красная пешеходная сигнализация (см. рисунок 4.17).

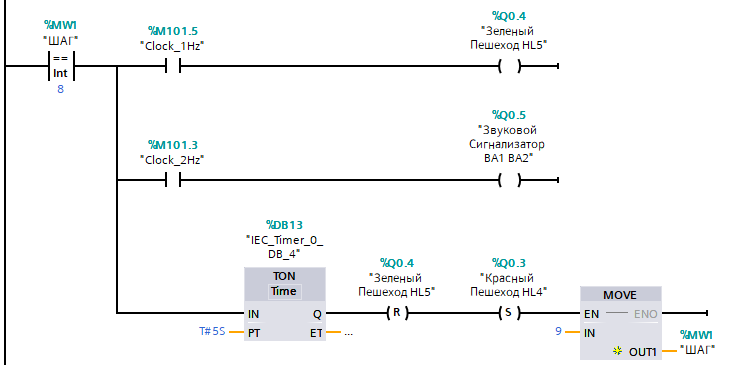


Рисунок 4.17 – Мигание зеленой пешеходной сигнализации (*Network 10)*

Переменной (ШАГ) присваивается значение девять, и программа переходит на следующий шаг.

Когда переменная (ШАГ) равняется девяти, включается желтая транспортная сигнализация (см. рисунок 4.18).

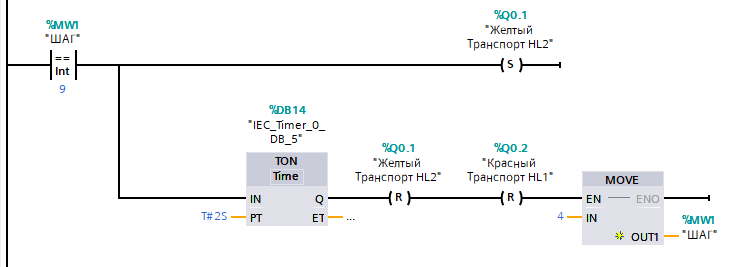


Рисунок 4.18 – Включение желтой транспортной сигнализации (*Network 10)*

После двух секунд выключается желтая и красная транспортные сигнализации и переменной (ШАГ) присваивается значение четверки, программа переходит на шаг номер четыре.

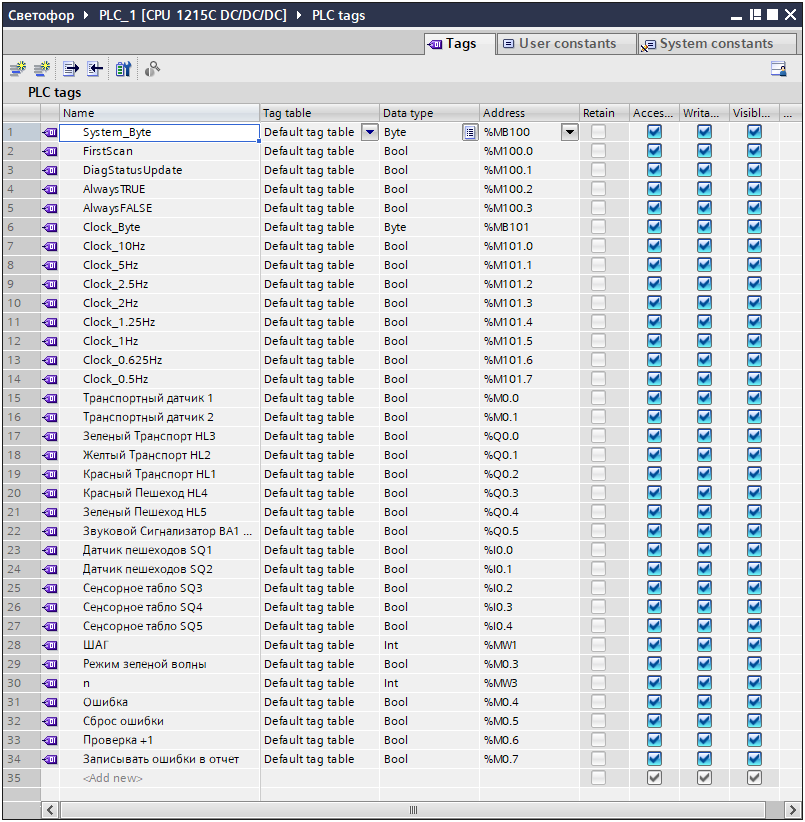


Рисунок 4.19 – Окно *Default tag table*

Наименование тегов отражают их технологическое назначение и указывают на связь с контроллером. В таблице тегов указывается наименование переменной, тип данных переменной, адрес переменной, комментарии к переменной.

Полный код программы на языке *LAD* приводиться в приложении.

Таким образом, в разделе разработана программная часть проекта. Для реализации использована интегрированная среда разработки *TIA Portal* 15*.* Средствами данной среды реализована программа ПЛК *SIMATIC S*7-1200. Для создания программы использовался язык релейных схем *LAD*.