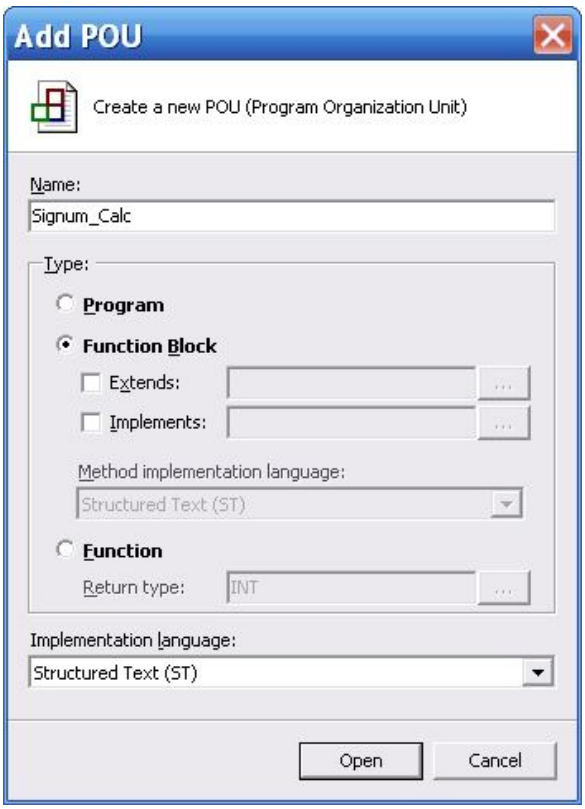
Упражнение 22. Часть 2. Управление перемещением

1. Управление перемещением с постоянной скоростью

1.1 Откройте *в SoMachine* проект под названием "*22_1 Encoder*" с ПЛК *M258*, сохраните проект под именем "*22_2 Encoder task*".

1.2 Добавьте переменные *CMD* (слово управления состоянием преобразователя частоты), *REF_ATV* (задание скорости вращения привода), *Pos_fact* (текущее значение положения рабочего органа, определенное с помощью энкодера) и *Pos_ref* (заданное перемещение) в список глобальных переменных, сохраняя их типы из предыдущего проекта.

1.3 Дополните созданное приложение функциональным блоком для вычисления знака скорости вращения, выполнив команду *Application / Add Object / POU / Function Block*.

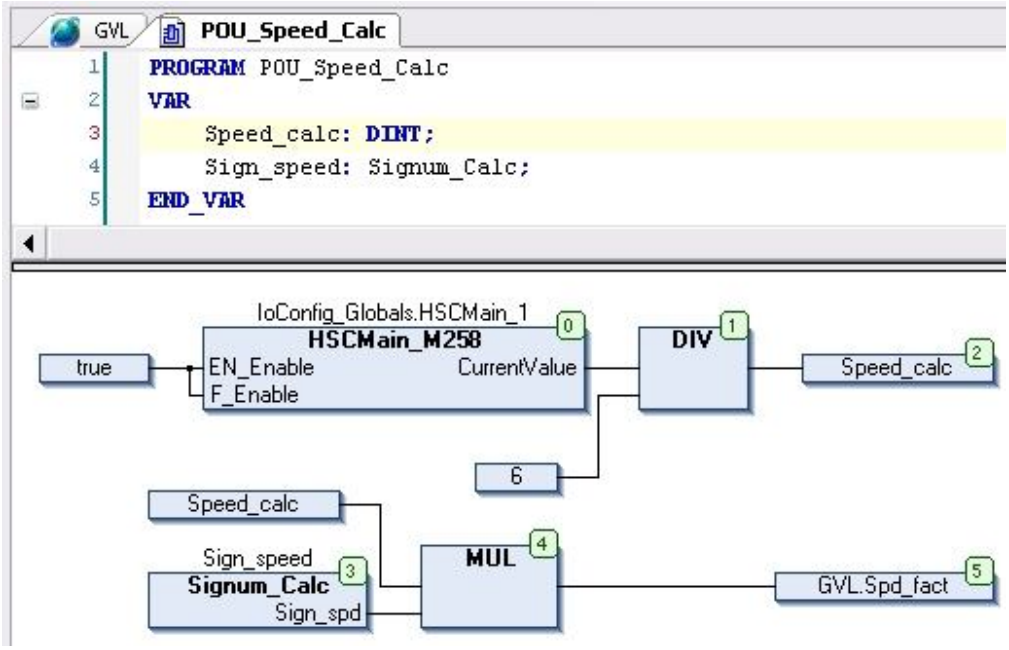


Созданный функциональный блок сконфигурируйте по примеру

```
1  FUNCTION_BLOCK Signum_Calc
2  VAR_INPUT
3  END_VAR
4  VAR_OUTPUT
5      Sign_spd: INT;
6  END_VAR
7  VAR
8  END_VAR
9
```

```
1  IF CAN_SPD>=0 THEN
2      Sign_spd:=1;
3      ELSE Sign_spd:=1;
4  END_IF;
```

Выход функционального блока може принимать значения «1» или «–1» в зависимости от направления движения привода. Используйте далее созданный функциональный блок для дополнения модуля вычисления фактической скорости с быстрым счетчиком *POU_Speed_calc*. (Для компактного отображения функциональных блоков выберите кнопку  «*Remove Unused Pins*»). Пример программы вычисления фактической скорости приведен ниже.

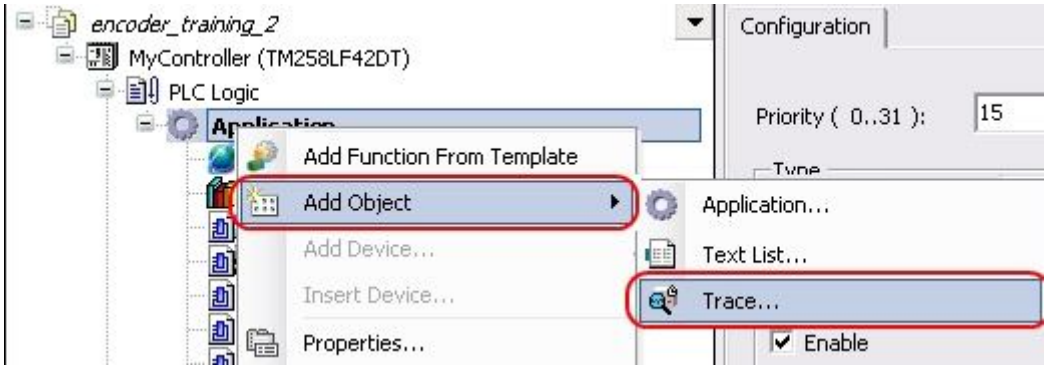


1.4 Добавьте в узел *Application* дерева проекта новый программный модуль *POU_Motion_Task* с языком программирования *ST* (*Structured Text*). Создайте код программы, который предусматривает выполнение перемещения вала привода при рассогласовании текущего положения с заданным. Пример программы с переходом на пониженную скорость для точной остановки в заданном положении приведен ниже.

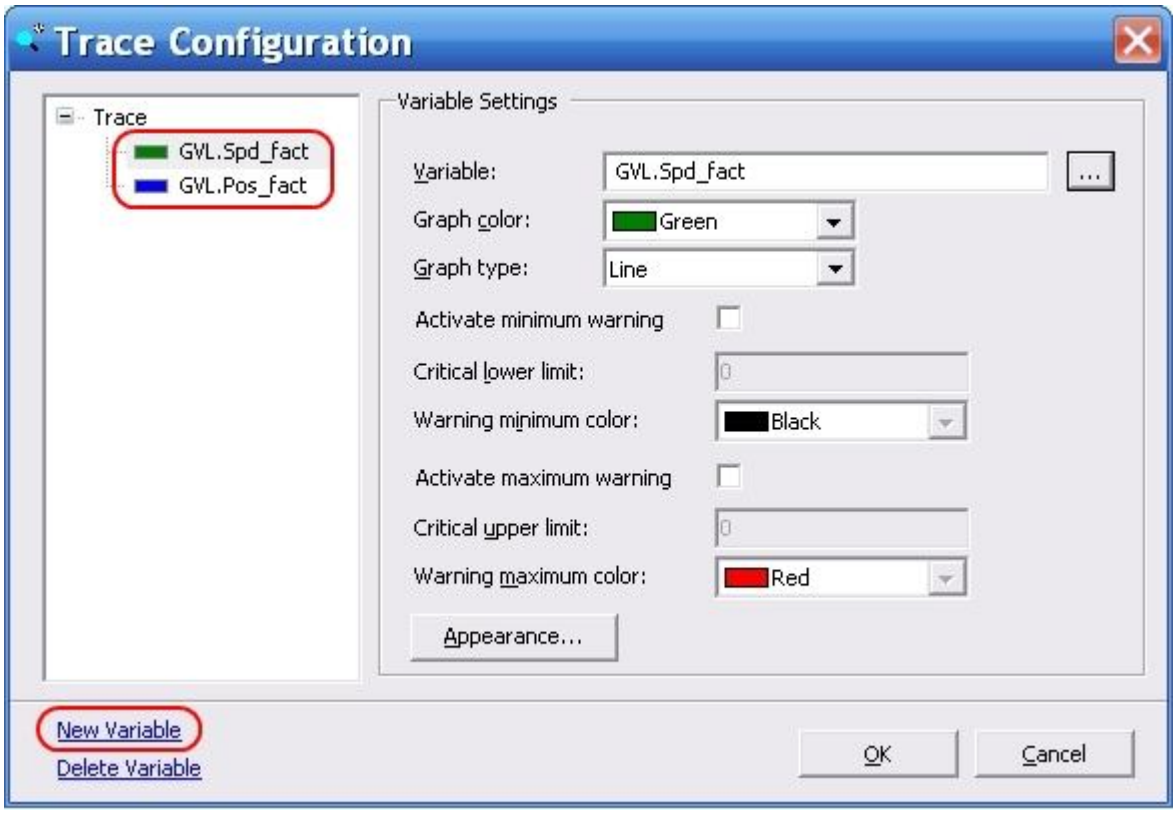
```
POU_Motion_Task
PROGRAM POU_Motion_Task
VAR
    Ab_Pos: DINT;
END_VAR

Ab_Pos:=ABS(GVL.Pos_ref-GVL.Pos_fact); // вычисление текущего рассогласования положения
GVL.CMD:=0;
IF GVL.Pos_fact<GVL.Pos_ref THEN
    GVL.CMD:=1;                    // вперед, если положение меньше задания
END_IF;
IF GVL.Pos_fact>GVL.Pos_ref THEN
    GVL.CMD:=2;                    // назад, если положение больше задания
END_IF;
IF Ab_Pos<40 THEN
    GVL.REF_ATV:=GVL.Spd_ref/6;    // переход на пониженную скорость перед остановкой
    ELSE GVL.REF_ATV:=GVL.Spd_ref;
END_IF;
```

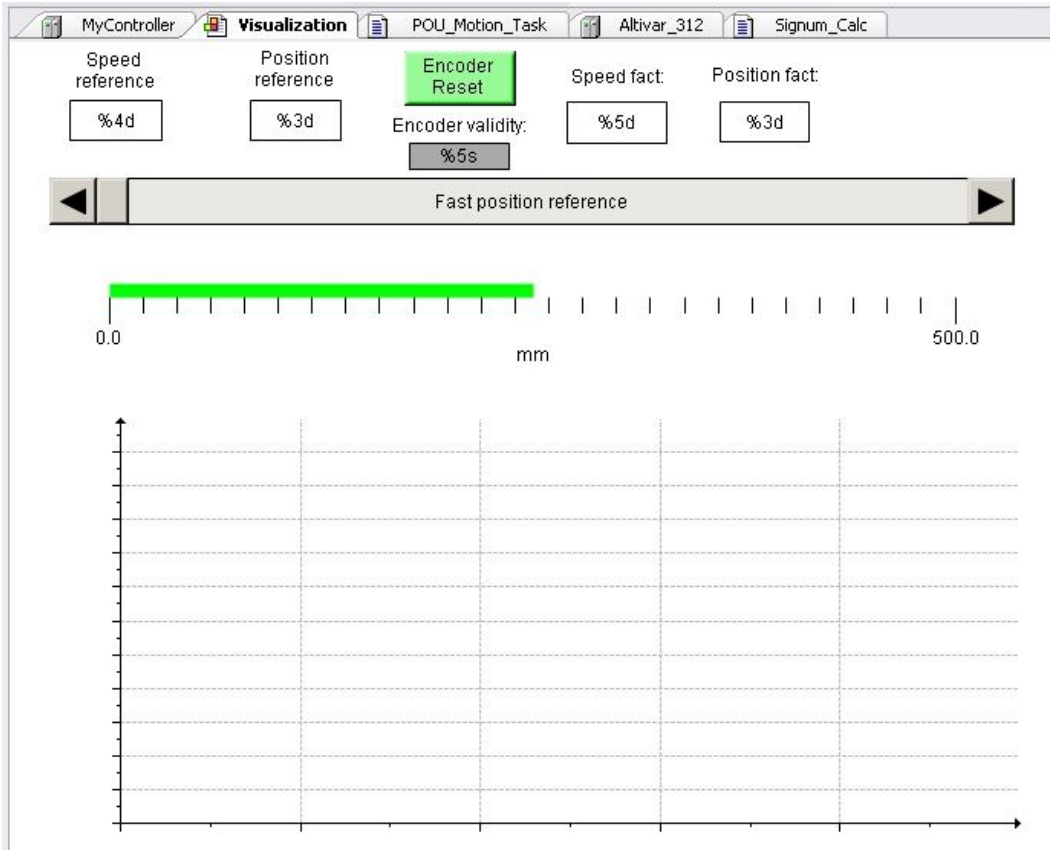
1.5 Добавьте в узел *Application* дерева проекта ветку ведения трендов перевенных *Trace*.



Настройте отображение в тренде значений переменных текущей скорости (*Spd_fact*) и текущего положения вала оси (*Pos_fact*), использовав команду *New Variable*.

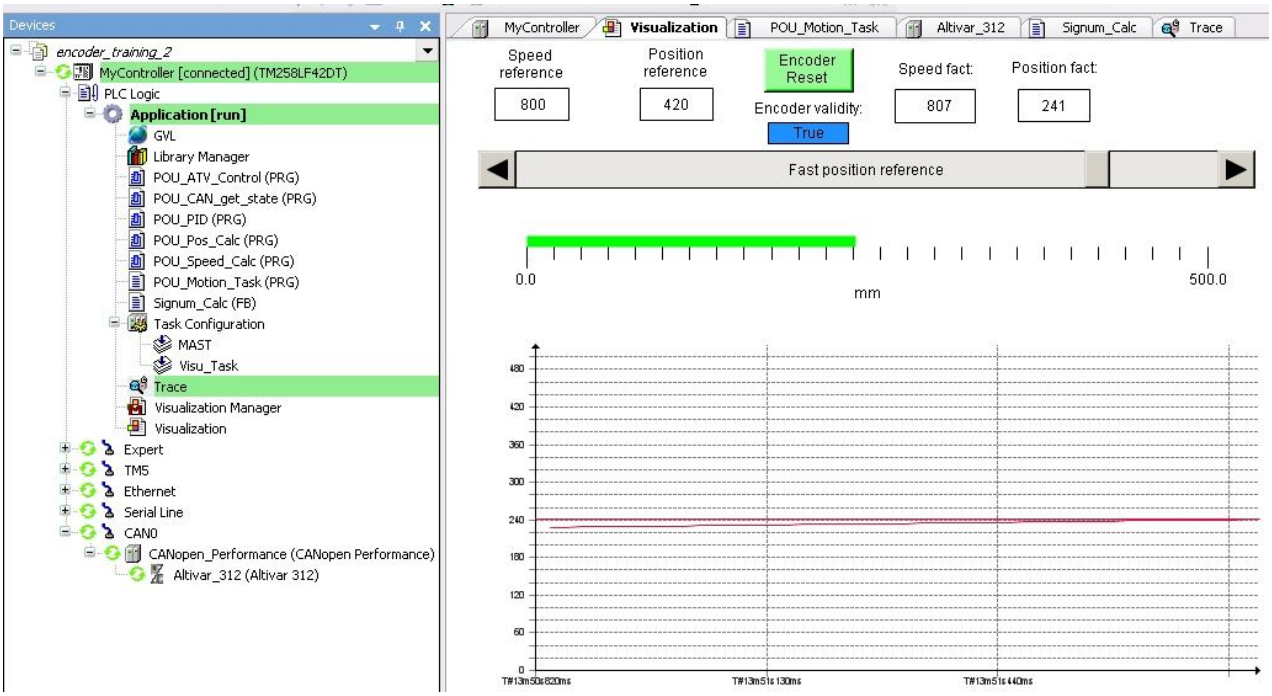


1.6 Создайте страницу визуализации проекта *Application / Add Object / Visualization.* На странице создайте графические объекты типа *Trace, Bar Display* из библиотеки *Complex Controls*, а также *Scrollbar* из библиотеки *Windows Controls.* Привяжите созданные объекты к переменным тренда, текущего положения и заданного положения соответственно. Создайте графические элементы для отображения и ввода задания по скорости и положению, сброса значения энкодера. Пример страницы визуализации приведен ниже.

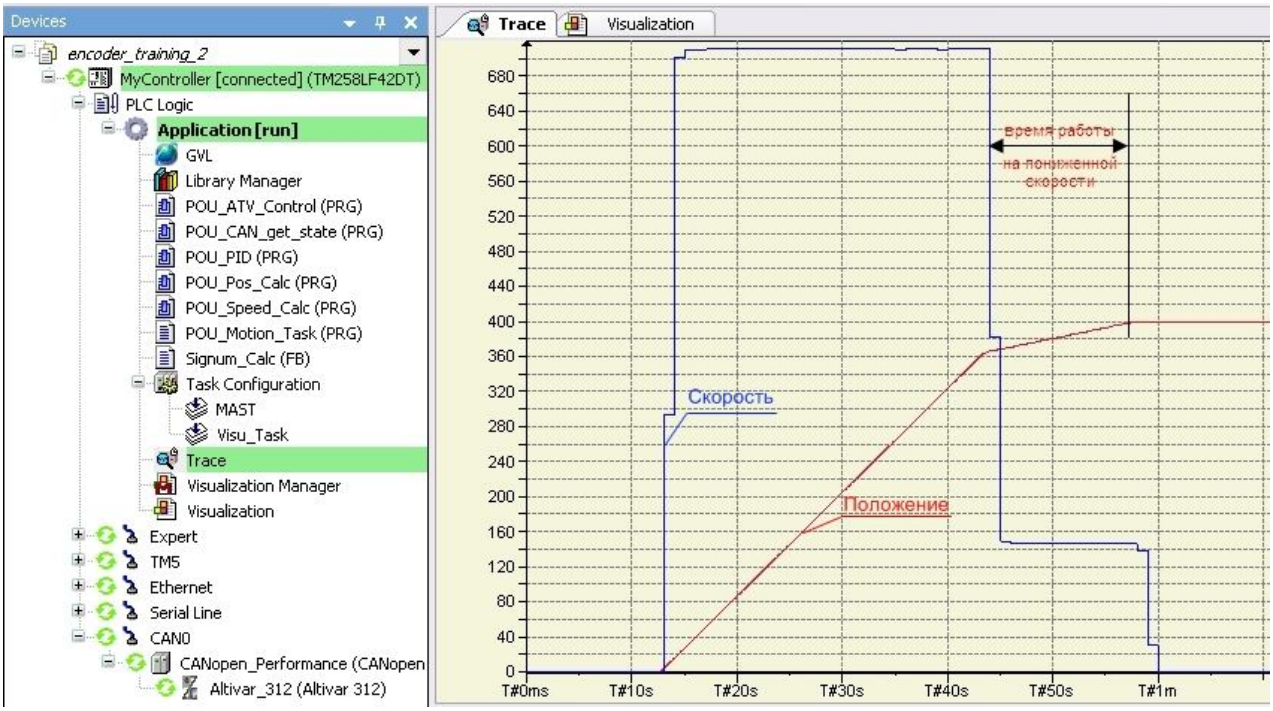


1.7 В узле проекта *Task Configuration* создайте новую задачу *Visu_task* (на 1 уровень приоритета ниже задачи *MAST*) и свяжите с ней ведение трендов переменных в окне *Trace Configuration*.

1.8 Скомпилируйте проект и передайте приложение в ПЛК. Проверьте работу приложения управления перемещением в режиме *On-line*, используя страницу визуализации, изменяя значения уставки положения и скорости.



1.9 Перейдите на страницу трендов *Trace* и выполните команду *Download Trace* с помощью кнопки  . Проследите за изменением скрости и положения в время отработки перемещения.



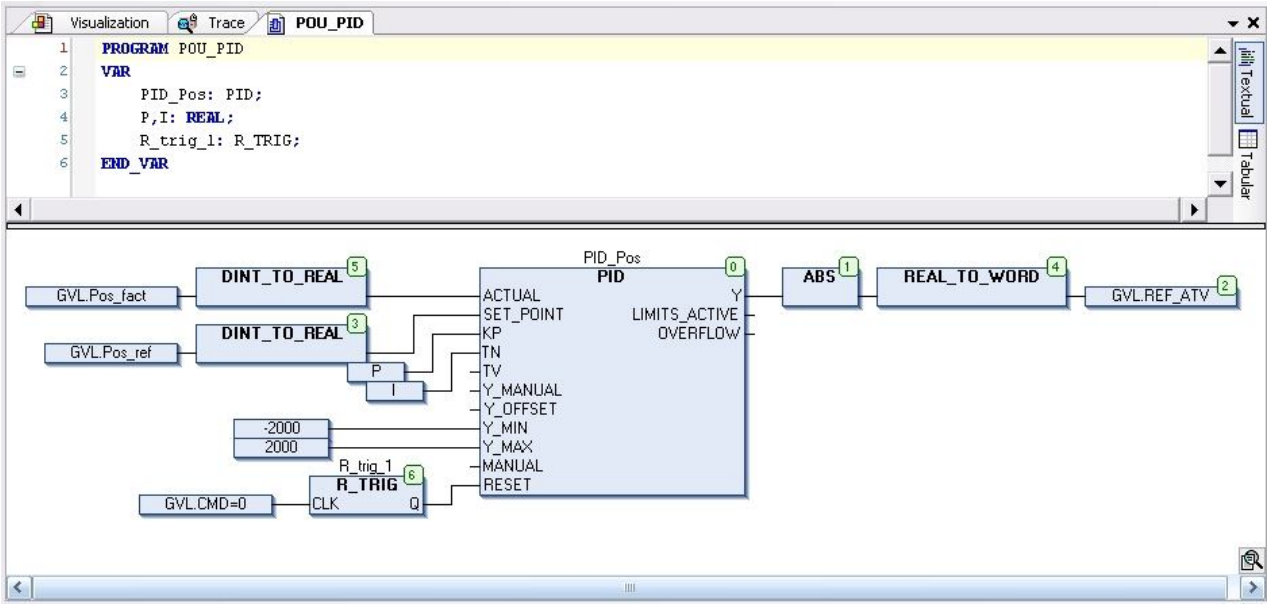
1.10 Сохраните проект, создав его резервную копию. Сделайте выводы по работе.

2. Управление перемещением с регулятором и обратной связью по положению

2.1 Откройте *в SoMachine* проект под названием “*22_2 Encoder task*” с ПЛК *M258*, сохраните проект под именем ”*22_3 Encoder task_PID*”.

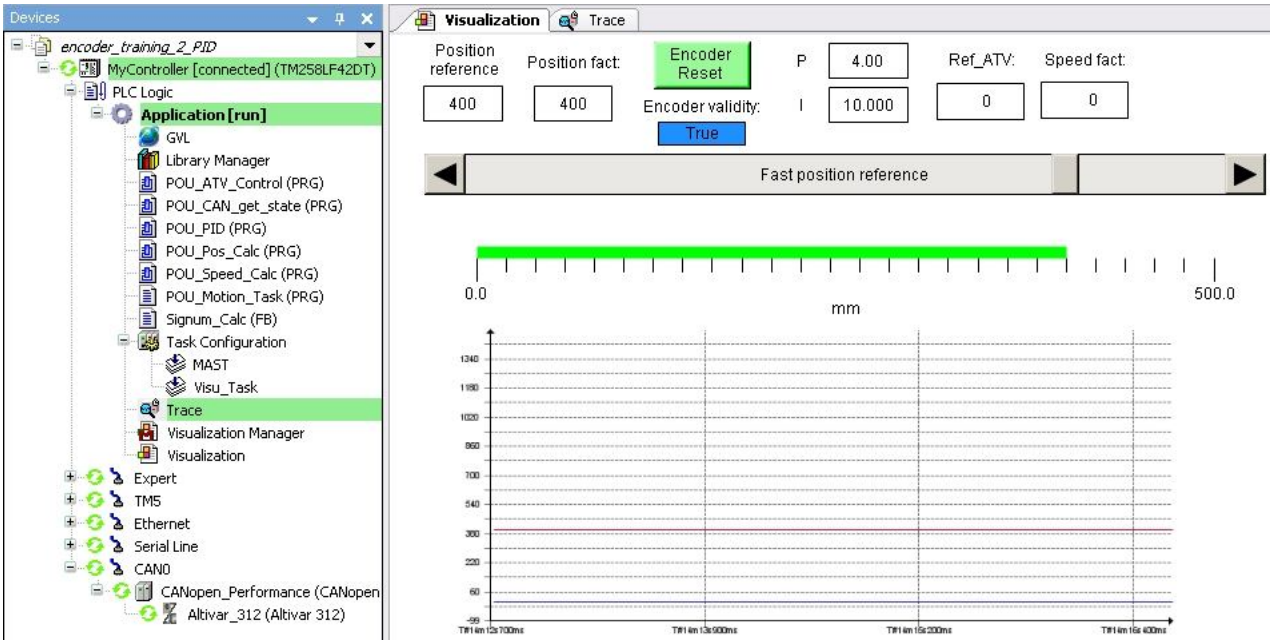
2.2 Измените программу *POU_Motion_Task*, удалив из нее условия изменения скорости (строки 9 – 12).

2.3 Добавьте в узел проекта *Application* новый программный модуль *POU* на языке *CFC*, назовите его *POU_PID*. Добавьте созданный модуль в список задач *MAST*. Создайте приложение модуля на базе блока *PID*, установив в качестве переменной обратной связи текущее значение положения, а в качестве выходной переменной – задание скорости перемещения привода. Предусмотрите автоматический сброс выхода *PID* при полной отработке перемещения. Пример программы приведен ниже.

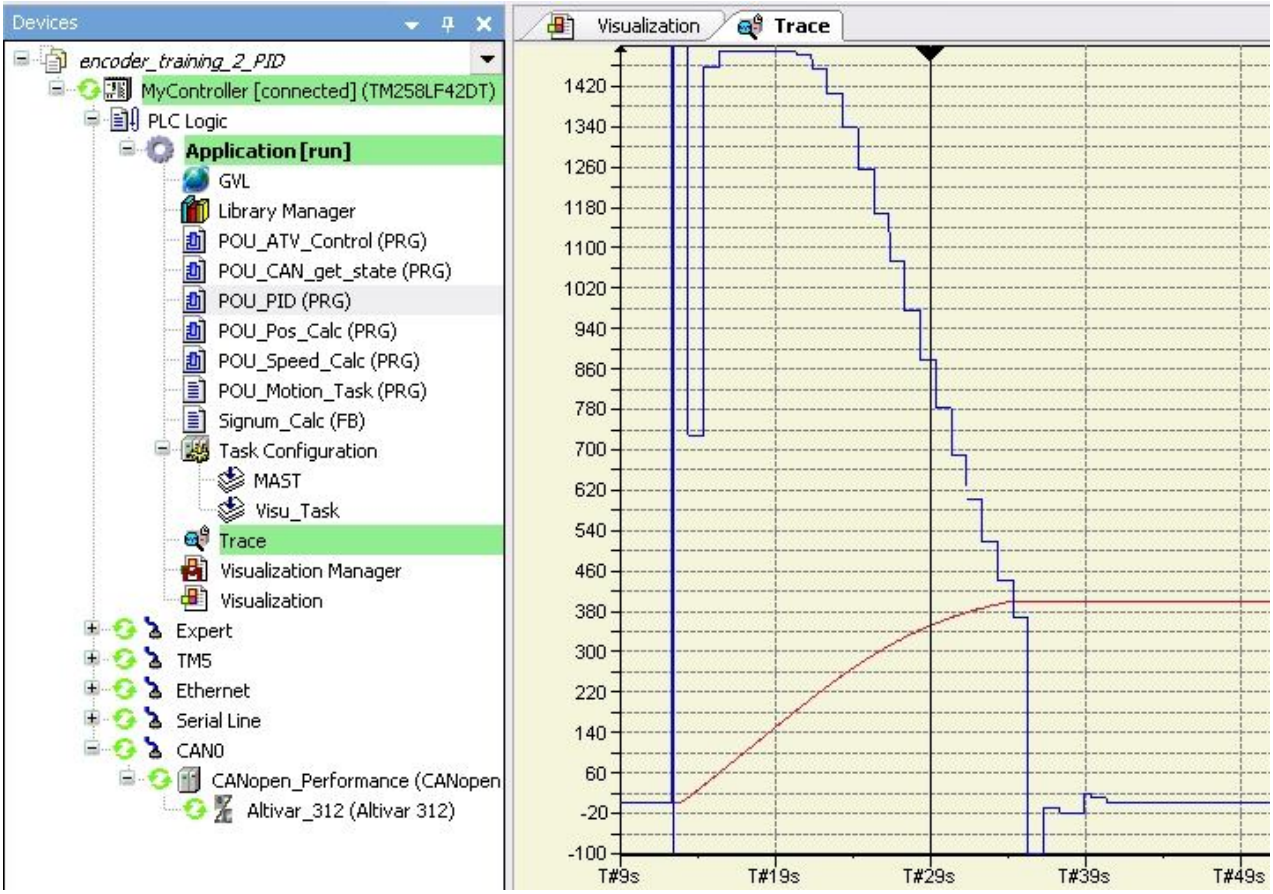


2.4 Измените страницу визуализации, заменив ячейку ввода заданной скорости ячейками ввода параметров ПИД-регулятора положения: *P* (коэффициента усиления) и *I* (постоянной времени интегрирования, с).

2.5 Скомпилируйте проект и передайте его в ПЛК. Проверьте работу программы управления перемещением в режиме *On-line* с помощью страницы визуализации и трендов.



2.6 Изменяйте параметры настройки регулятора положения для получения необходимых динамических характеристик системы: скорости перемещения, точности и времени регулирования.



2.7 Сохраните проект, создав его резервную копию. Сделайте выводы по работе.