**Автоматное программирование живет и медленно побеждает 2.**

Предлагаю вариант реализации программы *MetaAuto* С.Ю. Канжелева, сконфигурированную для языка *ST*, для практического применения при программировании ПЛК на *CoDeSys* или другой среде.

**Предыстория:**

Давно хотел применить в своей работе *SWITCH*-технологию, и вот, с переходом в газовую область появилась реальная возможность и необходимость использования *SWITCH*-технологии для написания программ управления всем, что связано с газом.

Для написания автоматной части программы использовались графы переходов автоматов, нарисованные в *Visio*.

Текст программы был сгенерирован из *Visio* при помощи конвертера *Visio2Switch* (Головешин А.)

**Проблема:**

На выходе *Visio2Switch* - C++ код => преобразуем его в *ST*. Я использовал *openc2pas* (http://c2pas.sourceforge.net/) + ручная доработка большего объема.

Переделать *Visio2Switch* под язы *ST* у меня не получилось, и я обратил внимание на аналоги, и остановился на *MetaAuto* С.Ю. Канжелева.

**Решение:**

В *MetaAuto*, для переделки выходного языка не нужно было лезть в исходники – достаточно исправить конфиги. Что и было сделано.

**Нюансы:**

В ходе работы был выявлен ряд особенностей этой программы (вероятно, связанных с «заточенностью» под этот http://is.ifmo.ru/works/evsys/ вариант *SWITCH*-технологии), но с большинством из них можно смириться.

Один из первых и критичных моментов – программа не запускалась без установки *Visual Studio.net 2003 Prerequisites*, которая регистрирует библиотеку *msvcp71.dll*.(возможно есть другие способы решения этой проблемы)

Другие особенности программы изложены в «_Инструкция по запуску.doc» из архива программы.

***SWITCH*-технология в ПЛК:**

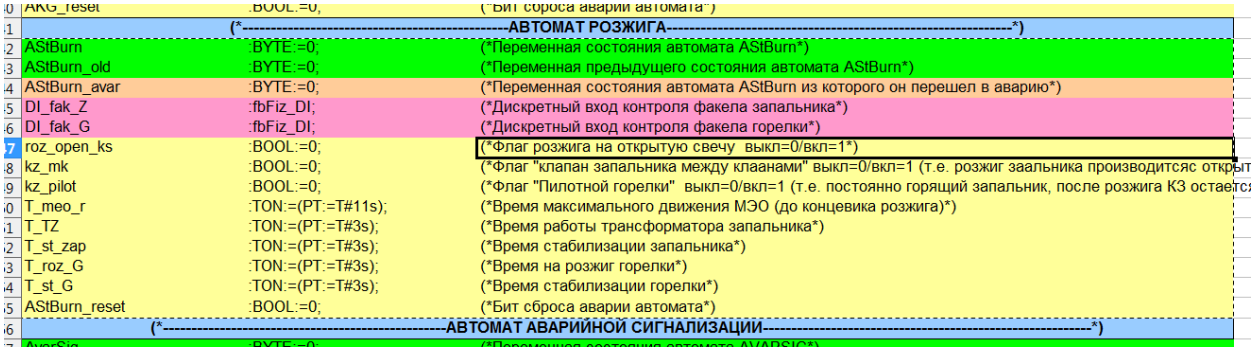
Основное, что хотелось бы сказать по поводу моего понимания *SWITCH*-технологии, применительно к программированию ПЛК это:

1. Один автомат – один *CASE*.

2. Нет понятий вложенность, вызываемость, событие. Все автоматы выполняются в каждом цикле ПЛК. Любое "событие" – это изменение переменной, ее и отражаем на графе. Взаимодействие автоматов – по номеру состояния.

3. Использовать реальные имена переменных, а не абстрактные *x5*, *e20*, *z35*. Улучшается читаемость и понимаемость текста. Описание переменных

не на графе, а в экселе, которые целиком копируются в кодесис и являются частью кода.



```
AKG_reset          :BOOL:=0;          (*Бит сброса аварии автомата*)
(*----------АВТОМАТ РОЗЖИГА----------*)
AStBurn            :BYTE:=0;          (*Переменная состояния автомата AStBurn*)
AStBurn_old        :BYTE:=0;          (*Переменная предыдущего состояния автомата AStBurn*)
AStBurn_avar       :BYTE:=0;          (*Переменная состояния автомата AStBurn из которого он перешел в аварию*)
DI_fak_Z           :fbFiz_DI;         (*Дискретный вход контроля факела запальника*)
DI_fak_G           :fbFiz_DI;         (*Дискретный вход контроля факела горелки*)
roz_open_ks        :BOOL:=0;          (*Флаг розжига на открытую свечу  выкл=0/вкл=1*)
kz_mk              :BOOL:=0;          (*Флаг "клапан запальника между клаанами" выкл=0/вкл=1 (т.е. розжиг заальника производитсяс открыт
kz_pilot           :BOOL:=0;          (*Флаг "Пилотной горелки"  выкл=0/вкл=1 (т.е. постоянно горящий запальник, после розжига КЗ остаетс
T_meo_r            :TON:=(PT:=T#11s); (*Время максимального движения МЭО (до концевика розжига)*)
T_TZ               :TON:=(PT:=T#3s);  (*Время работы трансформатора запальника*)
T_st_zap           :TON:=(PT:=T#3s);  (*Время стабилизации запальника*)
T_roz_G            :TON:=(PT:=T#3s);  (*Время на розжиг горелки*)
T_st_G             :TON:=(PT:=T#3s);  (*Время стабилизации горелки*)
AStBurn_reset      :BOOL:=0;          (*Бит сброса аварии автомата*)
(*----------АВТОМАТ АВАРИЙНОЙ СИГНАЛИЗАЦИИ----------*)
```

**Структура кода автомата на ST:**

Каждое состояние автомата состоит из разделов:

1. (*Действия при первоначальном входе в это состояние*) (необязательный)
2. (*Действия, постоянно выполняющиеся в этом состоянии*) (необязательный)
3. (*Запускаем таймер этого состояния*) (необязательный)
4. (*Запоминаем текущее состояние перед выполнением перехода*) (необязательный, если нет предыдущих)
5. (*Проверка условий на дугах и выполнение переходов*) (обязательный)

```
1: (*------------------------- Запускается -------------------------*)

IF AVEN_old<>AVEN THEN (*Действия при первоначальном входе в это состояние*)
        T_Run(IN:=FALSE);
        DO:=TRUE;
END_IF;                                                              1

(*Действия постоянно выполняющиеся в этом состоянии*)
        T_Run(IN:=TRUE);                                             2
(*Запускаем таймер этого состояния*)                                 3

AVEN_old:=AVEN; (*Запоминаем текущее состояние перед выполнением перехода*)  4

(*Проверка условий на дугах и выполнение переходов*)
                                                                     5
IF  TK_dvig THEN
    AVEN := 4;
ELSIF (Davl_para.D>=Davl_zaver_progrev) OR T_Zaver_progr.Q OR  NOT (Progrev) THEN
    AVEN := 0;
ELSIF ((PDS) AND ( NOT (T_Run.Q))) THEN
    AVEN := 2;
ELSIF  NOT (Run) THEN
    AVEN := 3;dhsrh;
ELSIF (( NOT (PDS)) AND (T_Run.Q)) THEN
    AVEN := 4;
END_IF;
```

Форум на эту тему находится по адресу http://www.owen.ru/forum/showthread.php?p=74642.

Ведмедь Антон
инженер КИПиА ООО «ПроектГазЭнергоНаладка».

P.S. `gmail` не разрешает передавать файлы с `exe` расширениями, поэтому архив запоролен. Пароль 1».