И.В. ПЕТРОВ (Компания “Пролог”)

Отладка прикладных ПЛК программ в CoDeSys (часть 1)

На сегодняшний день все большая ответственность за качество работы автоматизированных систем управления возлагается на прикладное программное обеспечение ПЛК. В то же время, языки программирования МЭК 611131-3 ориентированы не столько на профессиональных программистов, сколько на специалистов в прикладной области. Сами языки МЭК содержат ряд свойств, позволяющих сводить вероятность ошибок к минимуму. Это строгий контроль типов, специальные программные скобки, отказ от указателей, неявного преобразования типов и языковых конструкций, имеющих побочные эффекты. Высококачественные системы прикладного МЭК программирования помимо традиционных отладочных средств имеют не мало специфических инструментов. Некоторые рекомендации специалистов по отладке ПО не применимы для ПЛК физически. В данном цикле статей мы постараемся кратко рассмотреть как общие, так и специальные приемы создания надежного программного обеспечения для ПЛК, способы тестирования и поиска ошибок.

Материал базируется на средствах комплекса МЭК программирования CoDeSys, компании Smart Software Solutions GmbH (3S). На сегодняшний день CoDeSys применяется для программирования ПЛК более чем 70 европейских изготовителей. Достаточно высока вероятность того, что в практической работе вы столкнетесь именно с CoDeSys. Данный комплекс оснащен очень широким отладочным инструментарием, часть из которого характерна для всех МЭК систем программирования. Поэтому рассмотренные приемы будут полезны при работе с другими инструментами.

Специальные системные библиотеки CoDeSys позволяют переключить весь контроль работы ПЛК на уровень прикладной МЭК программы, в случае необходимости. Это включает контроль времени выполнения задач, прямую работу с аппаратными ресурсами и файловой системой, обработку аппаратных прерываний и исключений системы исполнения, управление планировщиком задач, синхронизацию задач, доступ к структурам конфигурации распределенных модульных систем и многое другое. Данные инструменты рассчитаны на специалистов, уникальны для CoDeSys и не доступны в других инструментах. Поэтому мы начнем с самых простейших вещей, постепенно углубляясь в более сложные детали.

Поскольку всегда лучше не допускать ошибок в программах, чем бороться с ними, мы выделим ряд проверенных практикой приемов надежного программирования на языках МЭК. Изложение проиллюстрировано примерами, выполненными в CoDeSys. Для выполнения абсолютного большинства примеров вам будет достаточно встроенного эмулятора ПЛК и демонстрационных версий SoftPLC (для Windows XP/NT), включенных в стандартный дистрибутив CoDeSys. Сама среда программирования CoDeSys поставляется бесплатно. Поэтому ни каких демо-версий CoDeSys не существует. Вы можете загрузить полный стандартный дистрибутив с сайтов www.codesys.ru или www.3s-software.com.

В первой части материала мы краткого напомним, как работать в CoDeSys, как писать и отлаживать простейшие МЭК программы. Эти навыки необходимо иметь, прежде чем двигаться дальше. Мы будем использовать CoDeSys версии V2.3 с английским интерфейсом.

Пишем простейший пример

Итак, CoDeSys успешно установлен на вашем компьютере и запущен. Создадим новый проект. В меню *File* дайте команду *New*. В появившемся диалоге выбора целевой платформы (тип ПЛК) *Target Settings* выберете *None*. Это означает, что мы будем пока работать со встроенным эмулятором CoDeSys. Нажмите кнопку *OK*. В следующем диалоге свойств нового программного компонента МЭК (POU) задайте тип POU: *Program*, имя нового компонента *PLC_PRG*. Эти параметры заданы по умолчанию, просто не изменяйте их сейчас. Мы обсудим их смысл позднее. В качестве языка программирования нового компонента выберете ST (См. рис 1). Нажмите кнопку *OK*. Мы создали новый, в прямом смысле слова, пустой проект.

В рабочем окне CoDeSys вы увидите окно редактора языка ST. Оно разделено на 2 части. Верхняя часть – это раздел объявлений. Нижний раздел – это собственно раздел реализации. Введите здесь текст нашей программы: `x := x+1;` и нажмите Enter. Наш проект практически готов.

 © НАУЧТЕХИЗДАТ 2006г.

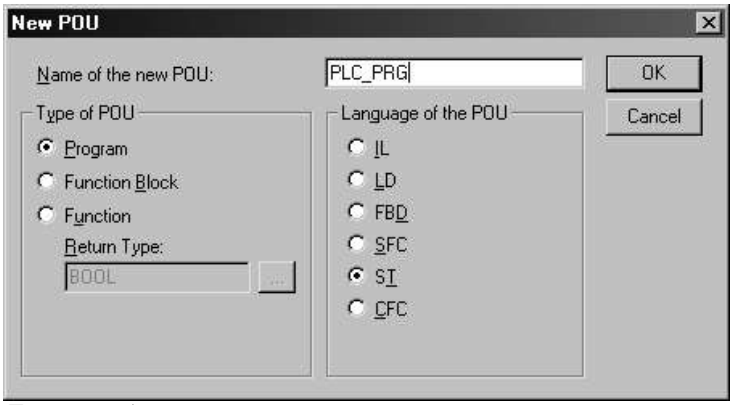

Рис.1. Диалог создания нового программного компонента

Основное внимание типам переменных

Языки МЭК относятся к языкам со *строгим контролем типов данных переменных*. В стандарте МЭК 61131-3 определен 21 тип данных. Правильный выбор типа данных позволяет писать надежные программы без излишних проверок. Применение базовых и пользовательских типов МЭК рассмотрено в [1]. Приемы работы с типами данных: длительность, время дня, дата и временной штамп весьма подробно рассмотрены в [2].

При вводе примера, мы не объявили переменную X и не указали ее тип. Это наша первая ошибка. К счастью она совсем проста, CoDeSys легко это понимает и автоматически предлагает диалог определения новой переменной (См. рис 2). Задайте тип переменной INT (введите тип вручную или нажмите кнопку […] и выберете тип из списка стандартных базовых типов). Выбор оптимального типа данных для переменных исключительно важен и мы еще вернемся к этой проблеме не один раз.

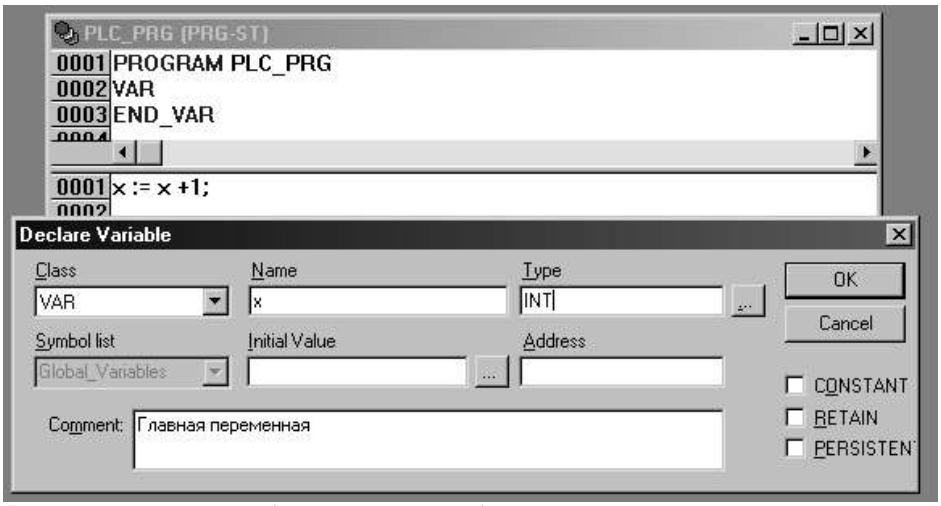

Рис. 2 Реализация примера и диалог авто-определения переменной

Проверяйте введенный текст сразу

Обратите внимание на раздел объявлений. Ключевые слова выделены здесь ярким синим цветом. Попробуйте в нашем примере вместо PROGRAM написать PROGRA. Окраска данного слова изменится на черный цвет. Так работает *синтаксическое цветовое выделение*. Это очень полезное и широко распространенное средство систем программирования. Оно позволяет замечать и исправлять синтаксические ошибки еще *до компиляции* программы. Цветовое выделение в CoDeSys работает во всех текстовых редакторах. По умолчанию, ключевые слова выделяются синим цветом, комментарии зеленым, явные ошибки красным, типизированные константы фиолетовым (например, T#1ms), неиспользуемые объекты (в менеджере проекта) серым цветом. Следите за цветовой окраской текста вашей программы при вводе, это поможет выявить многие опечатки сразу, не теряя времени на возврат к ним при отладке.

Сохраняйте результаты вашей работы

Пока мы не записали наш новый проект на диск, он не имеет имени (называется Untitled). Дайте команду *File, Save* и сохраните проект под некоторым осмысленным именем, например *Example_1*.

CoDeSys умеет восстанавливать открытый файл проекта, после экстренного выключения питания компьютера. В «больших» ПЛК высшего класса CoDeSys умеет сохранять исходный проекта в памяти контроллера. Дополнительный инструмент 3S: инжиниринговый сервер (ENI) позволяет контролировать версии всех изменений в вашем проекте и возвращаться к любому варианту при необходимости. С помощью встроенного инструмента документирования можно создать распечатку, по которой проект может быть полноценно восстановлен. В любом случае, о сохранности результатов работы нужно заботится пока они еще целы. Восстановить исходные тексты программ в приемлемом для нормальной работы виде (с правильными именами переменных и комментариями) из машинного кода нельзя.

Все что относится к одному проекту, CoDeSys хранит в единственном файле с расширением *prj*. Для полного сохранения результатов вашей работы достаточно сохранить единственный файл в надежном месте. Если вы не используете ENI, то не забывайте периодически делать резервные копии рабочих версий вашего проекта.

Компилируйте проект

К сожалению, в процессе редактирования текстов CoDeSys V2.3 умеет обнаруживать только очень примитивные синтаксические ошибки (в V3.0 это не так [3]). Для полной проверки необходимо выполнить компиляцию проекта. Воспользуйтесь командой меню *Project Build* или клавишей *F11*. Не нужно пытаться написать всю программу целиком и затем откомпилировать. Компилируйте проект после написания каждого более или менее целостного фрагмента. Ошибки лучше исправлять сразу. Вы можете забыть все тонкости задуманных конструкций: правильные имена переменных, соответствие открывающих и закрывающих скобок в выражениях. В итоге вам придется потратить больше времени на анализ и локализацию ошибки.

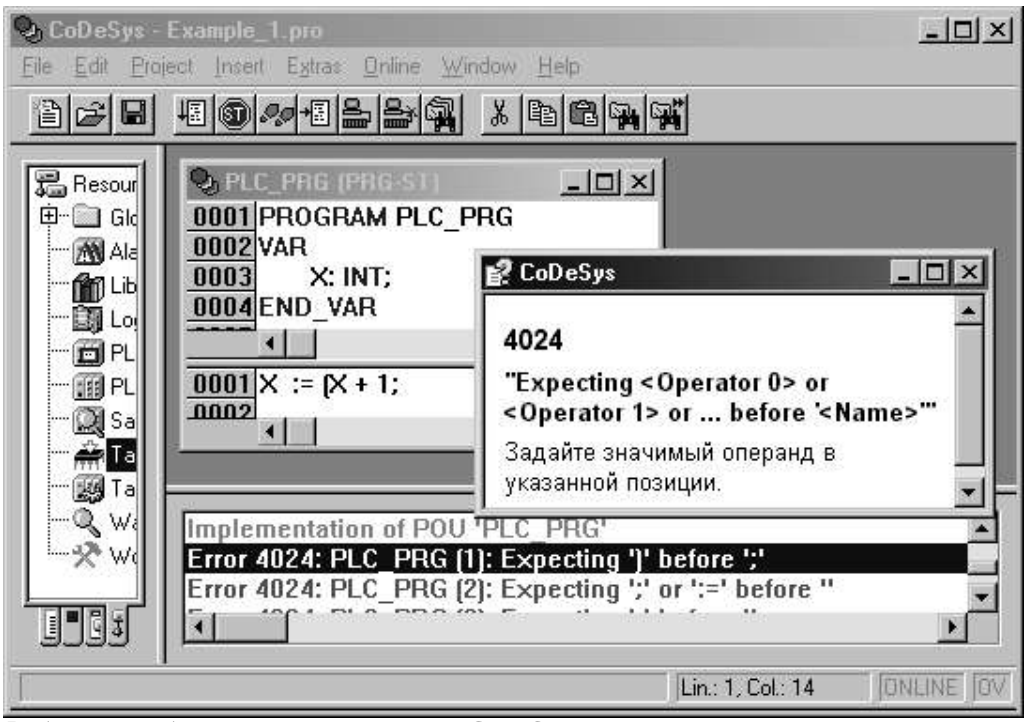

Рис. 3 Работа с сообщениями компилятора в CoDeSys

CoDeSys выводит результаты компиляции в специальном окне сообщений, расположенном в нижней части рабочего экрана (См. рис. 3). Лучшим результатом является: *0 Error(s), 0 Warning(s)*. Если вам не повезло, то перейдите в окно сообщений и найдите самое первое сообщение об ошибке. Щелчок мышки или нажатие *Enter* на тексте сообщения приведет к установке курсора на место в тексте вызвавшее проблему. Сообщения об ошибках всегда нужно анализировать

и исправлять начиная с первого. Очень часто одна опечатка может привести к нарушению ряда условий и вызвать целый букет сообщений.

Предупреждения (warnings) указывают на потенциально опасные моменты, которые не мешают нормальной компиляции проекта, но могут вызывать неоднозначности. Игнорировать предупреждения может только очень опытный программист, в особых случаях, при четком понимании того, что он делает. В общем случае, к предупреждениям нужно относится с тем же вниманием, как и к ошибкам. Измените сомнительную конструкцию так чтобы она «понравилась» компилятору. Предупреждения формируются на основе опыта и их наличие говорит о плохом стиле программирования.

CoDeSys оснащен быстрым инкрементальным компилятором. «Лишняя» компиляция не может повредить вашему проекту, но может сэкономить время отладки. Не забывайте о периодической проверке своей работы. Если в вашем тексте есть обращения к не существующим (пока не реализованным) POU, то сделайте пустые заглушки для них. В ST компонентах достаточно поставить ; в тексте, чтобы компилятор не выдавал предупреждения. Добейтесь полного отсутствия ошибок и предупреждений.

Если в процессе ввода программы вам мешает окно сообщений, уберите его с помощью быстрой комбинации клавиш *Shift-Esc*. При компиляции окно сообщений будет открыто автоматически.

Используйте контекстную помощь

Нажав Ctrl-F1 в CoDeSys вы получите контекстную подсказку. Это справедливо и для окна сообщений.

Попробуйте внести ошибку в текст нашего примера. Например, так: *X:=(X+1;*. Откомпилируйте проект (F11). Выбрав текст сообщения и нажав Ctrl-F1, вы получите более подробную информацию об ошибке (См. рис 3).

Чтобы получить помощь на русском языке (для V2.3.5 и старше), достаточно выбрать его в опциях проекта (Project, Options, Desktop, Language: Russian).

Отлаживаем простейший пример

Продолжим работу с нашим примером. Дайте в CoDeSys команду *Online, Login*. На панели управления этой команде соответствует иконка с изображением компьютера, связанного проводом с ПЛК. Если проект изменялся, то он будет автоматически перекомпилирован и загружен в память ПЛК. В нашем случае программа загружена в память эмулятора, но не запущена. Контроллер находится на связи с отладчиком (login) в режиме **стоп**.

Обратите внимание, что переменная X имеет значение 0. CoDeSys всегда выполняет начальную инициализацию переменных при перезагрузке или сбросе контроллера. Это гарантирует однозначность выполнения кода. Если вы не задали явно при объявлении начального значения переменной, то она будет обнулена.

Работа в режиме Стоп

В режиме стоп система исполнения CoDeSys опрашивает входы выходы контроллера и производит все обычные вспомогательные действия с единственным исключением – прикладная программа не выполняется.

Не смотря на то, что в режиме стоп программа ни чего не делает, это самый полезный режим при отладке. В режиме стоп все данные переменных «заморожены» и мы можем изучать их сколь угодно долго.

С помощью отладчика CoDeSys мы всегда можем остановить программу. Для этого существует команда *Online Stop* и соответствующая иконка на панели управления. Остановка программы выполняется не мгновенно в отличие от точек останова (мы рассмотрим работу с ними позднее) а, по окончании рабочего цикла ПЛК (определение рабочего цикла дано в [1]).

В реальном ПЛК мы имеем возможность доступа к значениям входов и выходов ПЛК. Это дает возможность проверить монтаж оборудования. Так для изучения работы датчиков мы можем формировать внешние воздействия и наблюдать результат в числовом формате или в виде графиков (графическая трассировка). Все это можно делать «написав» и загрузив в ПЛК пустую программу, содержащую только конфигурацию входов/выходов (для модульных систем).

Выполнение по циклам

В режиме стоп нажмите *Ctrl-F5* (команда *Online, Single cycle*). Контроллер отработает один очередной рабочий цикл и вновь остановится.

Выполняя код в реальном времени, мы можем воздействовать на входы и наблюдать результат. Но увидеть каким путем контроллер пришел в данное состояние, мы просто не успеем физически. Каждый рабочий цикл контроллера это один такт его работы. В каждом такте программа получает значения входов и вычисляет значения выходов. Выполнение по циклам дает нам возможность детально отследить все промежуточные действия программы. Выполнение проекта в отладчике по рабочим циклам является самым наглядным способом отладки логики работы программ для ПЛК.

Практическая реализация отладки по циклам в системе исполнения не требует ни каких дополнительных ухищрений, наподобие вставки специальных отладочных кодов. Система исполнения просто останавливает контроллер после выполнения рабочего цикла. В контроллере работает в точности тот код программы, который будет исполняться в реальности.

Для просмотра значений переменных в CoDeSys не нужно делать ни чего. Непосредственно в редакторе, в котором вы вводили программу, вы уведете текущие значения переменных. Вы можете просмотреть значения всех переменных в разделе объявлений редактора. В CoDeSys нет отдельного отладчика. В online редакторы соответствующих языков автоматически превращаются в окна отладчика. Для языка ST, в разделе реализации, справа от каждой строки показаны значения переменных задействованные в данной строке.

Пошагайте по рабочим циклам с помощью *Ctrl-F5*. Вы увидите как изменяется значение переменной X после каждого цикла.

Обратите внимание: значения переменных показанные справа от инструкций в режиме останов получены в конце полного рабочего цикла, а не сразу после выполнения данной команды! Об этом нельзя забывать. Иначе вы можете впасть в заблуждение и попытаться исправить правильно функционирующий код.

Давайте закрепим это на примере. Отключитесь от контроллера командой *Online, Logout*. В редакторе ST введите еще одну аналогичную строку: *X := X + 1;* Перейдите в *Online* и прошагайте несколько рабочих циклов. Посмотрите внимательно на рисунок 4. Везде где отображено значение переменной X, мы видим число 8. Совершенно очевидно, что после первой строки нашей программы переменная X имела значение 7. В режиме стоп мы видим значения всех переменных так, как их «видит» ПЛК после выполнения очередного рабочего цикла.

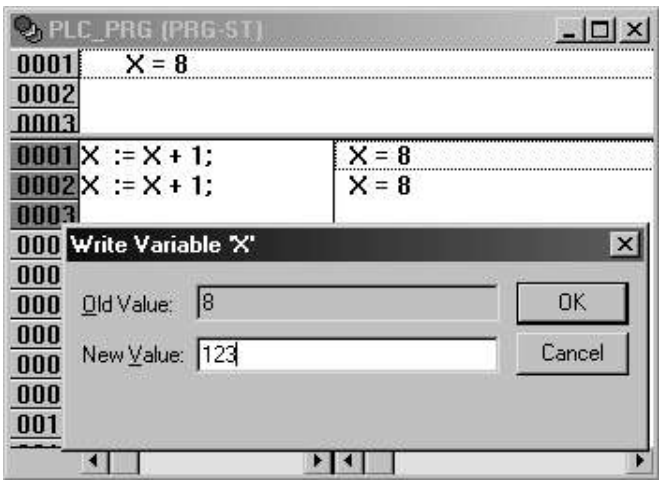

Рис. 4 Мониторинг и модификация значения переменной

Изменение значений переменных

На рисунке 4 мы видим значения переменных в десятичной системе счисления. Если вы щелкните по значению переменной правой клавишей мышки, то в контекстном меню можно будет выбрать шестнадцатеричную или двоичную систему счисления. Не забывайте об этой удобной возможности при работе с битовыми строками.

Двойной щелчок левой клавишей по переменной откроет диалоговое окно изменения значения переменной (См. рис 4). Однако введенное нами новое значение не записывается в ПЛК мгновенно, как следовало бы ожидать. Оно сохраняется только на экране среды программирования CoDeSys, выделенное голубым цветом в угловых скобках (См. рис 5). Для физической записи всех измененных значений необходимо дать команду *Online, Write values (Ctrl-F7).*

Такой прием необходим для обеспечения одновременности записи значений переменных, если мы меняем не одно, а сразу ряд взаимосвязанных переменных. Особенно важно это в режиме исполнения в реальном времени.

Выполнение в реальном времени

Дайте команду *Online Run (F5)*. Контроллер начнет выполнять программу так, как это будет происходить в реальных условиях. Отладчик позволяет просматривать и изменять значения переменных, как и в режиме стоп.

Здесь необходимо учитывать следующее: значения переменных считываются из памяти ПЛК с определенным интервалом асинхронно (обычно от 0.1 до 2х секунд, зависит от настройки системы). То есть для быстрых процессов мы будем видеть меняющиеся значения, по последовательности изменения которых нельзя судить о последовательности работы программы. В нашем примере, реальный контроллер за 100 мс может прокрутить рабочий цикл огромное число раз. Переменная X будет переполняться многократно. Мы же будем видеть только отдельные числа, выхваченные из этого непрерывного процесса.

Работа отладчика в CoDeSys практически не влияет на скорость выполнения программы. В 8 разрядных ПЛК (8051 22МГц) мониторинг 20 переменных типа INT увеличивает среднее время рабочего цикла контроллера менее чем на 2%. Отладчик, как и OPC сервер 3S используют единый механизм подключения. То есть работа SCADA системы скажется на ПЛК аналогичным образом. В 32х разрядных контроллерах с вытесняющей многозадачностью такое влияние отсутствует полностью (например, в CoDeSys SP RTE). При желании, вы можете написать программу, измеряющую время рабочего цикла ПЛК (используйте функцию Time()) и с помощью графической трассировки детально изучить влияние различных факторов на работу вашего ПЛК.

При выполнении программы в эмуляторе контроллер тактируется по системному таймеру ПК. То есть мы работаем с ПЛК имеющим время рабочего цикла около 55мс. Режим эмуляции в CoDeSys включается двумя способами. 1) Выбор *None* в качестве аппаратной платформы. 2) При работе с реальным ПЛК, в меню *Online* включите *Simulation mode*.

На случай если вам необходимо проверить определенные детали в ПЛК с гораздо более быстродействующей системой, вы можете использовать демо-версии контроллеров CoDeSys SP WinNT или CoDeSys SP RTE. Первый представляет собой быстродействующий 32х разрядный ПЛК с максимальной функциональностью. Второй контроллер реализует работу в жестком реальном времени с вытеснением задач под Windows XP/NT/2000. Оба варианта включены в стандартный дистрибутив. Демонстрационные версии имеют ограничение по времени непрерывной работы (2 часа). Конечно, можно использовать и реальный контроллер. Описанные отладочные инструменты будут работать точно также как и с эмулятором ПЛК. Маловероятно, что в начале работы с CoDeSys вам понадобится учитывать влияние быстродействия ПЛК и устройство его планировщика задач. Мы еще вернемся к этим вопросам. Сейчас же достаточно знать, что такая возможность есть.

Фиксация значений переменных

Измените значение переменной X, как описано выше (См. рис 4). Только теперь вместо команды *Online, Write values*, дайте команду *Online, Write values (F7)*. Значения переменной на экране будут выделены красным цветом. Это признак фиксированных значений переменных.

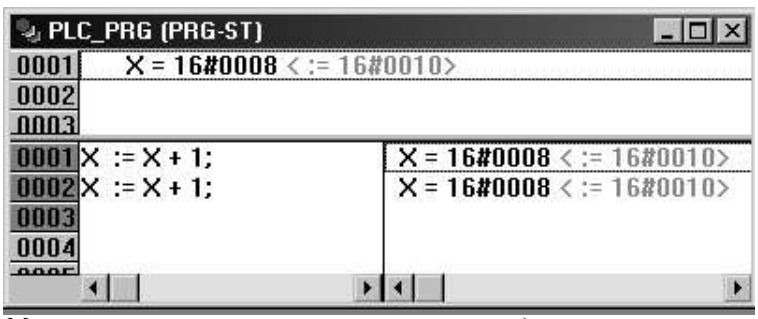

Рис. 5 Мониторинг значений переменных в шестнадцатеричной системе счисления

После выполнения программы в каждом рабочем цикле до записи значений на выходы, система исполнения подставляет значения фиксированных переменных вместо результатов вычислений программы.

Вы можете фиксировать любые переменные, в том числе входы и выходы ПЛК. Без такой возможности нам пришлось бы физически отключать определенные исполнительные механизмы, при отладке программы на реальном оборудовании. Фиксация выходов выполняет такое отключение программно. Вполне возможно что в реальной программе будут контролироваться значения выходов. Поэтому программная фиксация выходов даже более адекватна условиям отладки, чем аппаратное отключение.

Фиксируя значения входов, мы отключаем их от драйверов оборудования и можем моделировать внешние воздействия в режиме прогона в реальном времени.

Фиксация переменных программы позволяет моделировать необходимые ситуации или блокировать выполнение ветвей программ.

Список зафиксированных значений хранится в системе исполнения CoDeSys до отмены или сброса контроллера. Вы можете отключить CoDeSys от контроллера, фиксация будет работать. Для одновременной отмены всех зафиксированных значений используйте команду *Online, Release Force (Shift+F7)*.

Команда *Online, Write/Force Dialog* открывает диалог, включающий списки переменных, значения которых изменены (но не записаны) и фиксированных переменных. При отладке больших проектов, данный диалог удобно использовать для поиска всех установленных фиксаций и частичной их отмены.

Обратите внимание на статусную строку (внизу рабочего окна) CoDeSys в режиме online. Здесь указываются важнейшие режимы работы контроллера: режим эмуляции, фиксации переменных, запуска.

Наша первая задача имела вычислительный характер. Мы использовали текстовый язык ST. Описанные элементарные отладочные инструменты применимы для редакторов любого языка. Например, мы могли бы использовать язык список инструкций – IL. Если вам интересно, щелкните правой клавишей мышки по названию PLC_PRG в менеджере объектов CoDeSys (он расположен слева), в контекстном меню выберите команду *Convert Object*. В диалоге выбора языка выберите IL и нажмите OK. Наша программа будет автоматически конвертирована в IL. Результат показан на рисунке 6 (в режиме online с зафиксированным значением X).

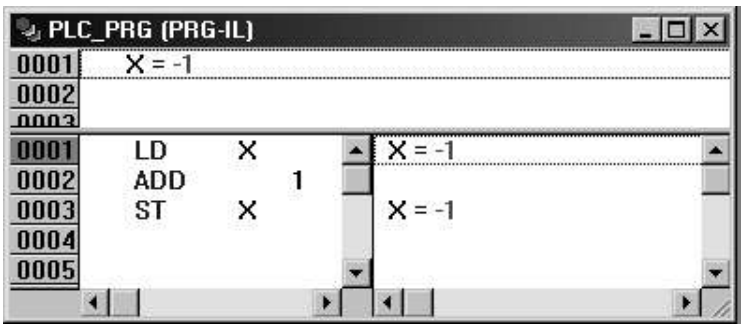

Рис. 6 Простейший пример на языке IL

Заключение

Мы рассмотрели самые простые, как модно говорить интуитивно понятные, инструменты отладки МЭК программ в CoDeSys. Теперь перейдем к более интересным вещам и более реалистичным примерам.

Продолжение следует.

Литература

1. Петров И.В. Программируемые контроллеры. Стандартные языки и приемы прикладного проектирования /Под ред. проф. В.П. Дьяконова. – М.: СОЛОН-Пресс, 2004.
2. Пастушенков Д.В., Петров И.В Программируем временные сложности. Промышленные АСУ и контроллеры. N7,8,9 – М.: НАУЧТЕХИЗДАТ, 2004
3. Петров И.В. CoDeSys 3.0 – новый уровень инструментов программирования ПЛК. СТА N2, 2005