В.В. ГРИБОВ, А.Я. КОРОСТЕЛЕВ, А.А. КОЙДА (ООО "Прайм Груп")

Тестирование программного обеспечения и отработка алгоритмов АСУТП для нефтегазового комплекса

Освещаются проблемы и методы отладки прикладного ПО (ППО) АСУТП, а также предлагается метод отладки ППО для ПЛК производства фирмы Siemens в комплексе с моделированием технологического процесса в среде Matlab.

The problems and methods of applied process control software debugging are outlined. The paper offers a debugging technique for applied software of Siemens's PLC complete with process simulation in Matlab.

Многие технологические процессы предприятий нефтегазового комплекса предполагают сложные многоконтурные системы управления технологическими объектами (ТО). При создании таких систем разработчику требуется находить ответы на множество вопросов: как связывать контуры системы управления, реализовывать каскадное управление или управление с коррекцией, какой алгоритм использовать для адаптивного управления. Часто возможны несколько вариантов регулятора, и требуется выбрать среди них тот, который обеспечивает наилучшее качество управления. Оценку можно производить с использованием специализированного ПО, например, в среде Matlab, где можно реализовать модель ТП и возможные варианты схем регулирования.

После выбора конкретной схемы регулирования и управления ИМ программист реализует соответствующие алгоритмы в ПЛК. На стадии отладки он сталкивается с проблемой проверки своей программы в условиях отсутствия технологического оборудования или невозможности активного эксперимента по соображениям безопасности. Даже если процесс управления не является сложным, в ППО возможны ошибки, которые проявляются на стадии пусконаладочных работ или, что еще хуже, в процессе эксплуатации объекта, что вызывает суммарное увеличение времени внедрения системы и, следовательно, влечет к лишним финансовым затратам. На поиск мелких ошибок в ППО порой уходит достаточно много времени, что не дает разработчику в полной мере сосредоточиться на законе управления. Возможны ошибки, связанные с взаимодействием ПЛК и верхнего уровня управления, или, например, такие ошибки управления, когда запускаются одновременно несколько алгоритмов по аварийным сигналам.

В принципе даже на объекте разработчик не может в полной мере проверить работу ППО, т.к. из-за временных ограничений и сложности ТО невозможно имитировать весь спектр штатных и нештатных ситуаций, тем более, если объект автоматизации является действующим. От разработчика АСУТП требуется полная уверенность в безошибочности написанного ППО, реализующего управление и контуры регулирования. Все это приводит к необходимости отладки и тестирования ПО на стадии проекта до начала пусконаладочных работ.

При комплексном тестировании системы необходимо обеспечить имитацию:

- режимов работы технологического оборудования;
- ТП (согласованное изменение параметров ТП);
- аварийных ситуаций.

Существует несколько методов тестирования ППО ПЛК. Самый простой способ отладки алгоритма – это включить имитацию ТО и ТП в программу ПЛК. Другими словами, входные/выходные переменные не привязываются к платам ввода/вывода, а изменяются из процедур и функций, имитирующих ИМ и режим работы технологической установки. Данный метод вполне пригоден для тестирования алгоритмов управления несложными ТО (кран, задвижка, клапан). Недостатком этого метода является то, что увеличивается размер ППО, и/или при установке ППО на объект требуется его коррекция, которая может привести к непредвиденным ошибкам.

При таком подходе разработчик имеет 2 комплекта программы, один из которых используется на полигоне, второй – установлен на объекте. Это затрудняет поиск ошибки в условиях отсутствия реального ТО и в тех случаях, когда требуется по описанию режима работы технологической установки найти причину неправильного управления. Также при таком подходе трудно имитировать сложные ТП, для моделирования которых необходимо решать дифференциальные уравнения, вводить задержки и т.д.

Другим методом тестирования ПО ПЛК является реализация работы ТО и режимов ТП в другом контроллере, который заменяет уровень датчиков и ИМ. Такой подход позволяет проводить комплексную проверку всей системы. Основными преимуществами такого подхода является то, что программа без изменения устанавливается на объекте, что модель объекта может разрабатываться независимо от управляющей программы. Недостаток этого метода – дороговизна, а также то, что не всегда разработчик имеет второй комплект ПЛК в своем распоряжении, и особенно – большое число модулей аналогового вывода, необходимых для имитации датчиков. При этом так же, как и при использовании предыдущего способа, возникает трудность реализации сложных ТП.

Ниже описывается метод тестирования ПО ПЛК для контроллеров производства Siemens сер. SIMATIC S7-300/400, для которых имеется эмулятор S7-PLCSIM. S7-PLCSIM эмулирует работу ПЛК на PC и позволяет отлаживать пользовательские процедуры и функции без наличия реального контроллера. Он обладает интуитивно понятным интерфейсом и имеет возможность вручную устанавливать значения сигналов ввода/вывода [1]. Но главным его преимуществом является то, что управление эмуляцией можно осуществлять из других приложений.

Для этого разработчики S7-PLCSIM предлагают пользователю ActiveX компонент S7ProSim, обеспечивающий доступ не только к области памяти ввода/вывода виртуального контроллера, но и к управлению циклом выполнения программы. Таким образом, модель ТП можно реализовать в любой программной среде, работающей под управлением Windows и поддерживающей технологию ActiveX.

Методы S7ProSim можно разделить на 3 группы:

- для установки и разрыва соединения с эмулятором S7-PLCSIM;
- для считывания и записи области ввода/вывода (управление платами УСО);
- для управления циклом программы в ПЛК.

Ниже кратко описываются методы взаимодействия пользовательской модели ТП с S7-PLCSIM, в котором загружено ППО управления ТП. Методы Connect и

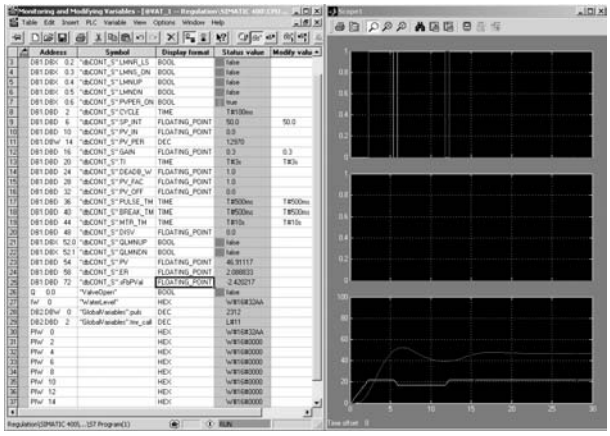

Рис. 1

Disconnect устанавливают и разрывают соответственно связь приложения с S7-PLCSIM по протоколу COM. Обе функции возвращают значение “0” при успешном выполнении или код ошибки соединения.

Метод ReadOutputImage(StartIndex, ElementsToRead, DataType, pData) считывает выходы ПЛК непрерывным блоком данных и возвращает код ошибки. StartIndex – начало области периферийных выходов, ElementsToRead – количество данных типа DataType, pData – область памяти, в которую будут возвращены значения выходных каналов ПЛК.

Метод ReadOutputPoint (ByteIndex, BitIndex, DataType, pData) в отличие от предыдущего позволяет узнать конкретный выход ПЛК, который однозначно определяется входными переменными ByteIndex (начальный байт), BitIndex (для дискретных выходов – бит в байте). Тип считываемого выхода определяется в DataType. При успешном выполнении функция возвращает ноль и значение выхода в pData, иначе – код ошибки.

Метод WriteInputImage (StartIndex, Data) переписывает область входных значений ПЛК, начинающуюся с StartIndex, значениями из области Data. При успешном выполнении функция возвращает ноль и значение выхода в pData, иначе – код ошибки.

Используя метод WriteInputPoint (ByteIndex, BitIndex, Data), пользователь может подать желаемую величину на вход ПЛК. Здесь ByteIndex определяет начальное смещение сигнала в области входов контроллера, BitIndex – бит (для дискретного входа), Data содержит записываемое значение.

Для управления циклом выполнения программы предназначены 3 метода: ExecuteNmsScan, ExecuteNScans, ExecuteSingleScan. Метод ExecuteNmsScan (MsNumber) инициирует выполнение работы программы в течение времени MsNumber, метод ExecuteNScans (NScanNumber) вызывает выполнение программы NScanNumber число раз, ExecuteSingleScan позволяет прогнать в виртуальном контролере один цикл ППО.

Более подробное описание методов и самого ActiveX компонента S7ProSim можно найти в документации [2], прилагаемой к нему, где приводится не только детальное описание S7ProSim, но также дается пример использования его компонента на Visual Basic (VB). Например, ActiveX компонент можно подключать в Excel, что позволяет накапливать статистическую информацию о работе программы и выводить ее в графическом виде, и, что самое главное, не требует никакой специализированной среды разработки (например, как Visual Studio).

На VB удобно реализовывать модель несложного ТП, имитировать нештатные и аварийные ситуации, регистрировать управляющие воздействия, но, когда речь заходит о процессах со сложными передаточными функциями, когда требуется имитировать запаздывание и решать дифференциальные уравнения, тогда время, необходимое для создания модели ТП, может намного превышать время, затрачиваемое на реализацию алгоритма управления. Для таких моделей гораздо удобнее использовать более развитые средства, предназначенные специально для моделирования сложных динамических систем.

Среди таких программных продуктов наиболее распространенным является Matlab, содержащий в своем составе пакет Simulink. С его помощью возможно моделирование линейных и нелинейных динамических систем, которые представляются функциональной блок-схемой. Simulink имеет обширную библиотеку компонентов и удобную среду разработки, что значительно сокращает время на разработку модели.

Далее рассмотрим возможность стыковки пакета Simulink в среде Matlab с эмулятором S7-PLCSIM через ActiveX компонент S7ProSim. При этом модель ТП строится с использованием развитых средств Simulink, а ППО эмулируется с помощью S7-PLCSIM (то есть можно отлаживать непосредственно ту программу, которая будет функционировать на объекте). На рис. 1 показана совместная работа Simulink и PLCSim.

Matlab имеет возможности связи с другими программами через протокол Component Object Model (COM). Следующие функции позволяют управлять S7-PLCSIM:

- h = actxserver (progid [, machinename]) – создает COM объект внутри Matlab;
- v = invoke(h, ['methodname' [, arg1, arg2, ...]]) – вызывает метод COM объекта;
- release(h) – освобождает интерфейс COM объекта или удаляет его.

Продемонстрируем предлагаемый метод отладки ППО на примере ТП, структура которого представлена на рис. 2.

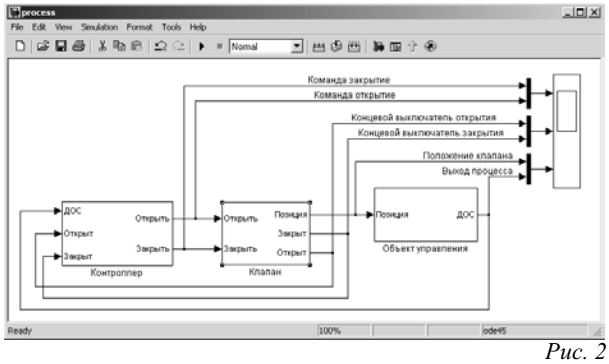

Рис. 2

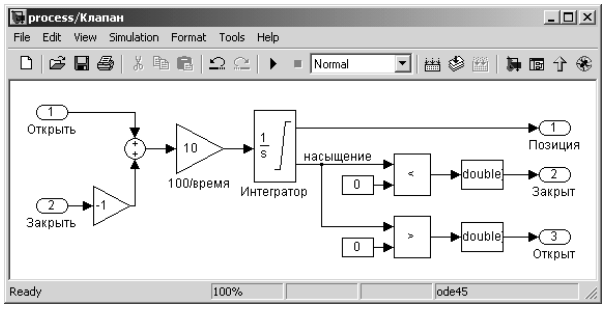

Рис. 3

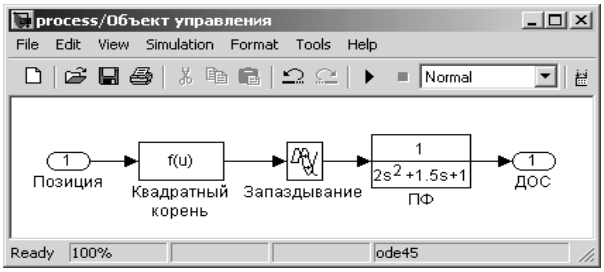

Рис. 4

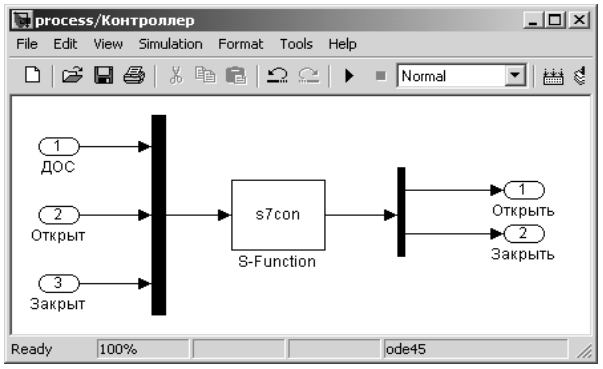

Рис. 5

Модель ТП состоит из трех подсистем: “клапан”, ”объект управления”, “контроллер”. В подсистеме “клапан” описывается ИМ (рис. 3). Подсистема “объект управления” имитирует ТП и включает в себя блоки: извлечения квадратного корня, запаздывания и передаточной функции $W(s) = \dfrac{1}{2s^2 + 1{,}5s + 1}$ (рис. 4). Подсистема “контроллер” осуществляет обмен данными с ППО, загруженным в эмулятор контроллера S7-PLCSIM (рис. 5).

Управляющее воздействие от контроллера (команды открыть/закрыть) поступает на вход подсистемы “клапан”, в которой вычисляется положение запорно-регулирующей арматуры. Далее величина, характеризующая положение клапана, поступает в “объект управления”, где вычисляется регулируемый параметр, который записывается в эмулятор ПЛК.

Обмен данными ввода/вывода между управляющей программой в ПЛК и моделью происходит в подсистеме “контроллер” и реализован в блоке, называемом *S*-функцией. Simulink позволяет с помощью *S*-функций, написанных на языке Matlab, создавать пользовательские блоки. Их можно использовать для реализации сложных алгоритмов, которые неудобно реализовывать с помощью функциональных блоков, входящих в стандартную библиотеку, а также для реализации взаимодействия с внешними программными (или даже аппаратными) средствами.

Входные и выходные параметры *S*-функции, а также ее внутренняя структура четко определены таким образом, чтобы система Simulink могла с ней работать. Проще всего создать собственную S-функцию, используя готовый шаблон sfuntmpl.m. Этот файл содержит заготовку *S*-функции, и пользователю достаточно лишь наполнить ее собственными алгоритмами.

S-функция на языке Matlab – это функция следующего вида:

[sys,x0,str,ts]=f(t,x,u,flag,p1,p2,...), где

- f – имя S-функции;
- t – текущее модельное время;
- x – вектор состояния пользовательского блока;
- u – вектор входов блока;
- flag – параметр, характеризующий действие, которое должна выполнить функция при данном вызове;
- p1, p2, .. – параметры пользовательского блока;
- sys – основной выходной параметр функции, который в зависимости от значения аргумента flag имеет разные значения;
- x0 – для нашего случая всегда пустой вектор;
- str – параметр, зарезервированный для использования в следующих версиях;
- ts – вектор, содержащий информацию о времени вызова функции (период вызова и смещение).

S-функция состоит из нескольких секций (mdlInitializeSizes, mdlGetTimeOfNextVarHit, mdlOutputs, mdlUpdate, mdlDerivatives, mdlTerminate). При каждом вызове выполняется только одна секция, которая определяется значением аргумента flag. Не будем описывать все правила написания *S*-функций, так как вся необходимая информация содержится в системе помощи Simulink, а остановимся лишь на некоторых моментах, которые имеют непосредственное отношение к описываемому методу отладки ППО.

Секция mdlInitializeSizes выполняется один раз в начале моделирования. Эта секция проводит инициализацию пользовательского блока. При этом заполняется структура sizes, которая содержит информацию о количестве входов, выходов, компонент вектора состояния блока, период вызова функции и т.д. В эту секцию пользователь может добавить любые действия, которые должны выполняться один раз в начале моделирования.

Структура sizes содержит следующие поля:

- sizes.NumContStates – количество непрерывных состояний (в данном случае 0);
- sizes.NumDiscStates – количество дискретных состояний (в данном случае 0);
- sizes.NumOutputs – количество выходов (в данном случае 2: команды открыть и закрыть);
- sizes.NumInputs – количество входов (в данном случае 3: выход процесса и 2 концевых выключателя, характеризующих крайние положения ИМ);
- sizes.DirFeedthrough – равен 1, если блок напрямую преобразует входы в выходы, то есть не имеет вектора состояния (как в нашем случае);
- sizes.NumSampleTimes – количество периодов вызова функции (в данном случае 1).

Для синхронизации работы модели и ПЛК необходимо, чтобы S-функция вызывалась через промежутки времени, равные циклу выполнения ППО в контроллере. Например, если цикл выполнения ППО в контроллере равен 100 мс, то осуществлять обмен информацией можно через этот интервал времени. Для управления периодичностью вызова S-функции служит выходной параметр ts. Вектору ts нужно присвоить вектор [0,1 0], это означает что функция будет выполняться каждые 0,1 с начиная с 0 с (время модельное). То есть Simulink будет вызывать функцию для расчета новых значений выходов блока в моменты времени 0, 0,1, 0,2, … с. и т.д.

Секция mdlOutputs служит для расчета выходных значений блока, вызывается на протяжении моделирования в моменты времени, которые были определены в секции mdlInitializeSizes.

Выходы блока рассчитываются следующим образом:

- сигналы, поступающие на вход блока, масштабируются и передаются в PLCSIM;
- выполняется один цикл программы контроллера;

• из PLCSIM считываются значения выходов и передаются на выход блока.

Секция mdlTerminate выполняется один раз при завершении моделирования. Пользователь может добавить в эту секцию действия, которые должны выполняться при завершении моделирования.

Ниже приводится текст используемой S-функции.

```
function [sys,x0,str,ts] = s7con(t,x,u,flag)

switch flag,
 % Инициализация
 case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
 % Вычисление выходов
 case 3,
 sys=mdlOutputs(t,x,u);
 % Не использованы
 case {1,2, 4, 9},
 sys=[];
 % Неизвестные значения
 otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
% end s7con

%=============================================
% mdlInitializeSizes
%=============================================
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 2; % 2 выхода
sizes.NumInputs = 3; % 3 входа
sizes.DirFeedthrough = 1; % блок Direct Feedthrough
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
% начальные условия
x0 = [];
% str всегда пустой вектор
str = [];
% S-функция будет вызываться каждые 100 мс. модельного времени
ts = [0.1 0];
% переводим PLCSIM в режим SingleScan
h = actxserver('S7wspsmx.S7ProSim.1')
invoke(h, 'Connect')
SetScanMode(h, 0)
invoke(h, 'Disconnect')
delete(h)
% end mdlInitializeSizes

%=============================================
% mdlOutputs
%=============================================
function sys=mdlOutputs(t,x,u)
%========== запись входов и выполнение одного цикла
comclient off;
h = actxserver('S7wspsmx.S7ProSim.1');
% соединение
invoke(h, 'Connect');
% масштабирование и запись входов
a=u(1)/100*27648;
invoke(h, 'WriteInputPoint', 0, 0, int16(a));
a=int16(1*(u(2)>0) + 2*(u(3)>0));
invoke(h, 'WriteInputPoint', 16, 0, a);
% выполняем один программный цикл
invoke(h, 'ExecuteSingleScan');
% завершаем соединение
invoke(h, 'Disconnect');
release(h);

%================ чтение выходов
comclient on;
h = actxserver('S7wspsmx.S7ProSim.1');
% соединение
invoke(h, 'Connect');
out = [0 0];
% чтение выходов
out(1)=ReadOutputPoint(h, 0, 0, 1, out(1));
out(2)=ReadOutputPoint(h, 0, 1, 1, out(2));
% завершение соединения
invoke(h, 'Disconnect');
release(h);

sys = out;
% end mdlOutputs
```

Надо заметить, что при использовании команды invoke (h, 'ExecuteSingleScan') время выполнения цикла не фиксировано. Например, цикл выполнения ППО сконфигурирован на 100 мс, и по команде invoke(h, 'ExecuteSingleScan') ППО закончило работу за 70 мс, тогда по контроллерному времени следующий цикл производится не через 100, а через 70 мс. В то же время обмен информацией с эмулятором происходит каждые 100 мс модельного времени. Это может привести к неправильному вычислению величин, зависимых от времени, например, неправильному интегрированию, вычислению производных, вычислению временных задержек и т.д. Поэтому, если в ППО используются функции вычисления времени между вызовами управляющих блоков, то их необходимо скорректировать таким образом, чтобы они возвращали фиксированное значение времени, соответствующее времени цикла ППО в реальных условиях.

Описанный подход тестирования ППО ПЛК SIMATIC S7-300/400 удобен для отладки структуры управления и анализа регуляторов ТП. Он предоставляет наглядную информацию о работе алгоритмов управления и позволяет разработчику отрабатывать ППО при отсутствии технологического оборудования. Данный метод тестирования и отладки ППО использовался при реализации алгоритмов управления и учета в составе измерительно-вычислительного комплекса (ИВК) “Прайм Искра” для коммерческого узла учета нефти (КУУН). ИВК “Прайм Искра” предназначен для автоматизированного измерения параметров сигналов, поступающих от первичных преобразователей и применяется для:

- определения массы, объема, температуры, давления, влагосодержания, плотности, вязкости нефти и нефтепродуктов на узлах учета, оснащенных турбинными, объемными, ультразвуковыми и массовыми преобразователями расхода;
- автоматизированного измерения уровня, гидростатического давления жидкости и определения массы нефти и жидких нефтепродуктов в градуированных емкостях (резервуары вертикальные и горизонтальные, транспортные емкости и т.п.);
- ведения учетно-расчетных операций при поставках нефти и нефтепродуктов.

Помимо информационного режима ИВК обеспечивает дискретное управление задвижками, а также регулирование давления на входе КУУН и скорости потока в линии качества.

Авторы будут благодарны за любые отзывы, предложения и замечания, которые можно направлять по электронной почте.

Грибов Виктор Валерьевич – руководитель группы программирования, Коростелев Александр Яковлевич – инженер-программист, Койда Александр Алексеевич – директор департамента АСУ и энергетики ООО “Прайм Груп”
Телефон (095)725-44-32.
E-mail:info@primegroup.ru
http://www.primegroup.ru

Список литературы

1. *SIMATIC*. S7-PLCSIM V5.0. User Manual
2. *SIMATIC*. S7ProSim. User Manual.
3. *MATLAB* Documentation.