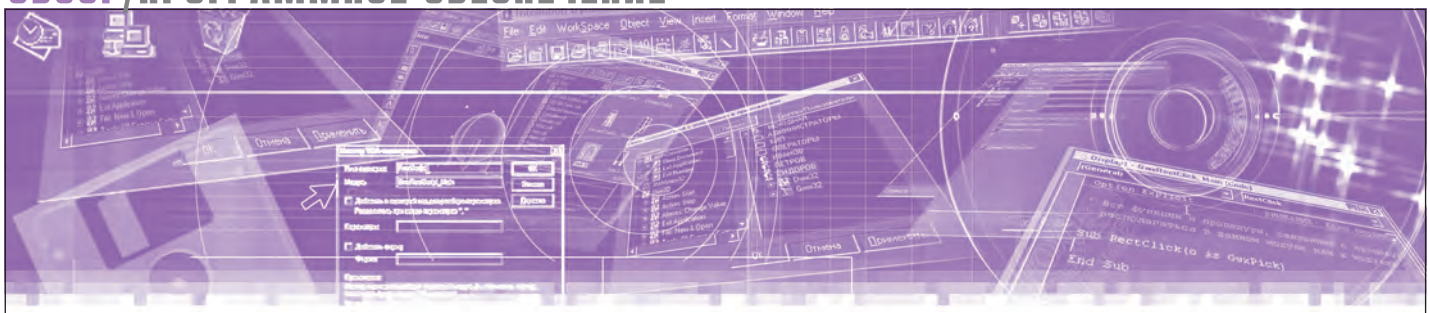

Сергей Федонин, Александр Наумов

Необычные компиляторы для необычных окружений

Необходимость миграции технологических процессов со старых зарубежных контроллеров на новые отечественные потребовала от разработчиков изучения устройства компиляторов промышленных языков программирования и сред разработки для них. Статья даёт представление о структуре и принципах построения данных компонентов, а также рассказывает о возможных путях оптимизации процессов миграции.

Введение

На фоне экономических санкций и заданного государством вектора на замещение программного и аппаратного обеспечения возникает вопрос: а как выглядит ситуация в контексте АСУ ТП? Точнее, в плане замены непригодных для дальнейшей работы иностранных контроллеров на их российские аналоги. К сожалению, в отличие от обычных персональных компьютеров, где процесс миграции с одной операционной системы на другую может осуществляться относительно просто, в том числе за счёт технологий виртуализации, в случае с переносом технологического процесса (ТП) с одной аппаратно-программной базы на другую дела обстоят менее радужно. Основные проблемы – излишняя закрытость существующих систем и отсутствие строгих стандартов на внутреннее устройство компонентов промышленных логических контроллеров (ПЛК). В этой статье рассмотрены методы решения одной из задач миграции ТП – перенос программ управления (ПУ) между контроллерами различных производителей. Стоит отметить, что исходный промышленный проект (ИПП), являющийся результатом миграции, отличается от ПУ тем, что предназначен не для исполнения в ПЛК, а для изменения разработчиком.

Первая часть статьи является ознакомительной, она описывает существующие подходы к построению программного обеспечения для языков стандарта МЭК 61131-3. В двух следующих частях изложен материал, необходимый для понимания того, какие данные об устройстве программного обеспечения производителей контроллеров нужны, чтобы успешно автоматизировать отдельные этапы процесса миграции ПУ ПЛК. В последней части в общих чертах представлены методы решения этой задачи. В качестве конкретного примера взяты контроллеры SIEMENS S7-300, SIEMENS S5-101U и REGUL R600 производства российской компании «ПРОСОФТ-Системы».

Обзор подходов к построению инструментальных средств для МЭК 61131-3

Современное программное обеспечение работает под управлением большого количества операционных систем на базе широкого спектра микропроцессорных архитектур. Одна и та же программа может существовать как для распространённой среди пользователей домашних ПК архитектуры IA-32 (Intel Architecture, 32-bit), так и для популярных в телекоммуникационном оборудовании архитектур MIPS и PowerPC. Тем не менее имеется целый класс программного обеспечения, исполняющегося в так называемых виртуальных машинах (ВМ). Они представляют собой модель реальной ЭВМ, которая обычно включает в себя микропроцессор, память и периферию. Но несмотря на важность каждого её компонента, центральное место в ВМ занимает виртуальный процессор. Фактически это программно реализованный конечный автомат, на вход которого подаются инструкции в закодированном виде (байт-коде), переводящие его из одного состояния в другое. Виртуальные машины в основном бывают двух типов – регистровые и стековые. В первом случае операции с данными выполняются через ограниченное количество ячеек памяти фиксированного размера (регистры), а во втором – через структуру данных типа стек. Известным примером программной платформы, использующей виртуальную машину для выполнения своего байт-кода, является Oracle Java. Не осталась без внимания данная технология и в программном обеспечении АСУ ТП, где она используется для исполнения ПУ на ПЛК. Наиболее важная и понятная причина выбора такого решения – переносимость. У производителя контроллеров может быть несколько несовместимых между собой линеек устройств, а в каждой из них – десятки различных версий используемой платформы. Поддерживать и совершенствовать средство генерации кода в таких условиях – процесс весьма трудоёмкий. В случае же с виртуальной машиной достаточно лишь перекомпилировать её исходные тексты под дру-

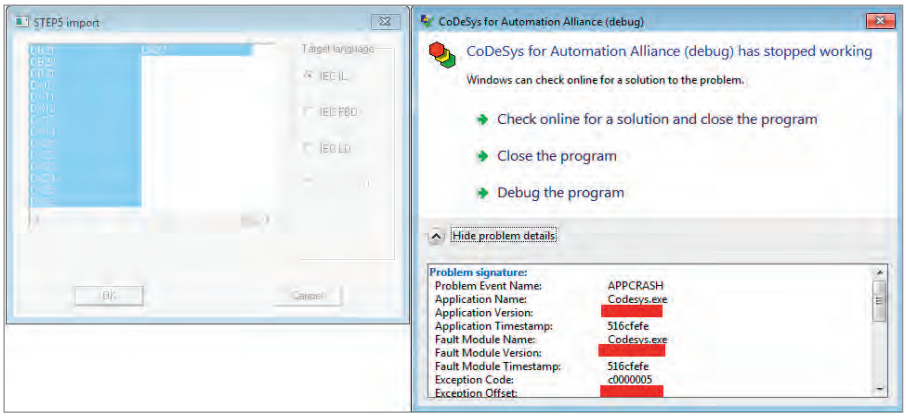

Рис. 1. Аварийное завершение среды CODESYS при попытке импорта проекта STEP 5

гую процессорную архитектуру, тем самым обеспечив поддержку исполнения программ управления без детального знакомства с ней. Впрочем, есть и другие, менее очевидные основания выбора ВМ для выполнения ПУ. Одно из них заключается в необходимости поддержки однозначного восстановления исходного кода программы из конечного представления, в которое она была скомпилирована. С этой целью набор виртуальных инструкций может быть спроектирован таким образом, чтобы он отображал особенности используемого промышленного языка программирования. При этом программист АСУ ТП видит только надводную верхушку айсберга – элемент меню Upload (загрузка ПУ из ПЛК) среды разработки, позволяющий ему получить проект с устройства в том (или почти в том) виде, в котором он туда устанавливался.

Отдельного упоминания заслуживает тот факт, что виртуальная машина контроллера иногда реализуется в аппаратном обеспечении, а не в коде встроенного программного обеспечения устройства. В первую очередь это делается для повышения скорости работы ПУ и предсказуемости времени выполнения отдельной инструкции, что особенно важно в связи со спецификой программного кода, управляющего ТП, – он должен работать строго в пределах определённого временно́го цикла. Фактически производитель размещает на основной микросхеме контроллера отдельный чип, содержащий в себе всю логику ВМ.

Несмотря на все плюсы использования описанной технологии, некоторые производители предпочитают компилировать промышленный проект в машинный код архитектуры, на базе которой работает вычислительный модуль. В большинстве случаев основная причина такого подхода – его дешевизна. Существует весьма популярная платформа CODESYS, предлагающая полный набор инструментария для компаний-производителей новых ПЛК. Чаще всего именно она берётся за основу в проектах, не имеющих основательных инвестиций на первоначальном этапе. Сделать программную начинку контроллера на ней существенно проще и быстрее, чем создавать все компоненты самостоятельно. Именно компилятор, входящий в её состав, и выдаёт на выходе двоичный код, напрямую выполняемый микропроцессором ПЛК. Главный недостаток такого решения – сложность реализации упомянутой ранее функции Upload, потому что однозначное восстановление исходного кода становится затруднительным в силу потери большого количества информации. Поэтому среда разработки, как правило, содержит специальную опцию, заставляющую компилятор добавлять тексты программы в бинарные файлы, устанавливаемые на устройство. Если её не активировать, то будет невозможно получить проект с ПЛК при его отсутствии на машине оператора.

Таким образом, о переносимости ПУ между различными контроллерами не может идти и речи. Особенно это касается тех пар ПЛК, программирование которых ведётся с помощью разных сред разработки, не имеющих общей кодовой базы. Всплывающая при этом проблема различия структур ИПП является не менее важной, чем проблема несовместимости сред исполнения ПУ. Выражается она, например, в невозможности открыть проект, разработанный в STEP 7 (для контроллеров SIEMENS SIMATIC S7-300 и S7-400), в среде Unity Pro (для контроллеров Schneider Electric Modicon). Это вынуждает разработчика АСУ ТП тратить дополнительные ресурсы на приобретение и изучение нужной среды разработки, даже если у него нет потребности в работе с контроллерами, ею поддерживаемыми. При этом стоит отметить, что некоторые производители добавляют возможность импорта ИПП других сред, чьими авторами они не являются. Так обстоит ситуация со средой CODESYS 2-й версии, позволяющей загружать в неё файлы проектов STEP 5 для ПЛК SIEMENS S5. Тем не менее эта функция пока несовершенна: помимо большого количества выдаваемых при конвертации ошибок, среда аварийно завершается на ряде реальных проектов (рис. 1). Кроме того, в 3-й версии CODESYS она отсутствует. Поэтому в целом нельзя сказать, что имеется устойчивая или хотя бы заметная тенденция к упрощению переноса ТП между объектами.

Устройство компиляторов промышленных языков компании SIEMENS

Достаточно популярные контроллеры компании SIEMENS линеек S5 и S7 базируются на собственном программном обеспечении, включая операционные

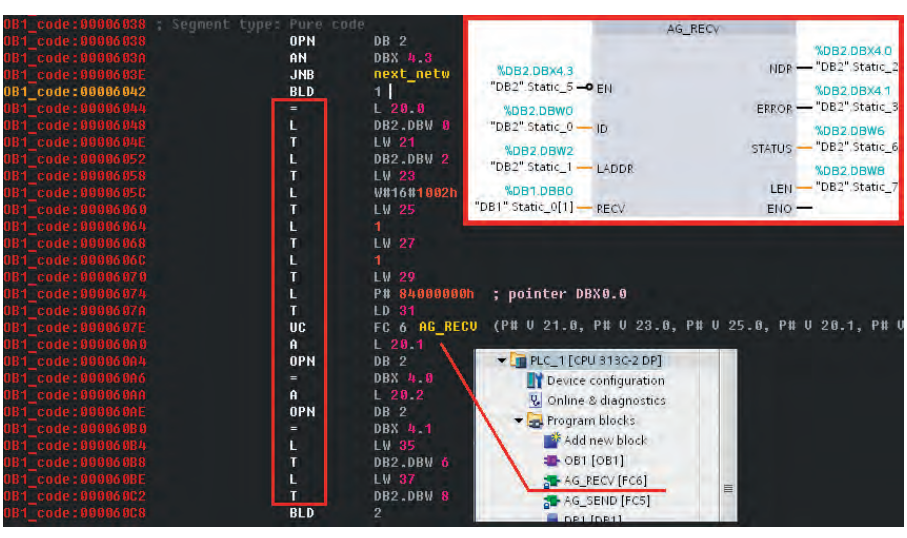

Рис. 2. Вызов AG_RECV (FC6) в исходном и скомпилированном (MC7) виде

```
OB1_code:00007070                   T       LW 29
OB1_code:00007074                   L       P# 84000000h
OB1_code:0000707A                   T       LD 31
OB1_code:0000707A ; ---------------------------------------------------------
OB1_code:0000707E         UC        .byte   3Dh ; =
OB1_code:0000707F                   .byte    0
OB1_code:00007080 ; ---------------------------------------------------------
OB1_code:00007080                   JU      save_outputs
OB1_code:00007080 ; ---------------------------------------------------------
OB1_code:00007084                   .dword 870000A8h
OB1_code:00007088                   .dword 870000B8h
OB1_code:0000708C                   .dword 870000C8h
OB1_code:00007090                   .dword 870000A1h
OB1_code:00007094                   .dword 870000A2h
OB1_code:00007098                   .dword 87000118h
OB1_code:0000709C                   .dword 87000128h
OB1_code:000070A0 ; ---------------------------------------------------------
OB1_code:000070A0
OB1_code:000070A0 save_outputs:
OB1_code:000070A0                   A       L 20.1
OB1_code:000070A4                   OPN     DB 2
```

Рис. 3. Передача параметров в AG_RECV

```
FB1_code:00003831 step_1:                             ; CODE XREF: sub_2281+DD8↑j
FB1_code:00003831                 JU      loc_3077    ; Jump Unconditional
FB1_code:00003835 ; ---------------------------------------------------------
FB1_code:00003835 ;  Network 14
FB1_code:00003835
FB1_code:00003835 step_2:                             ; CODE XREF: sub_2281+DDC↑j
FB1_code:00003835                 A       DIX 219.0   ; And
FB1_code:00003839                 S       Q 0.4       ; Set
FB1_code:0000383B                 JU      loc_3077    ; Jump Unconditional
FB1_code:0000383F ; ---------------------------------------------------------
FB1_code:0000383F ;  Network 15
FB1_code:0000383F
FB1_code:0000383F step_3:                             ; CODE XREF: sub_2281+DE0↑j
FB1_code:0000383F                 A       DIX 251.0   ; And
FB1_code:00003843                 S       Q 0.0       ; Set
FB1_code:00003845                 JU      loc_3077    ; Jump Unconditional
FB1_code:00003849 ; ---------------------------------------------------------
FB1_code:00003849 ;  Network 16
FB1_code:00003849
FB1_code:00003849 step_4:                             ; CODE XREF: sub_2281+DE4↑j
FB1_code:00003849                 A       DIX 283.0   ; And
FB1_code:0000384D                 =       Q 0.2       ; Assign
```

Рис. 4. Идущие друг за другом тела шагов (в терминологии GRAPH)

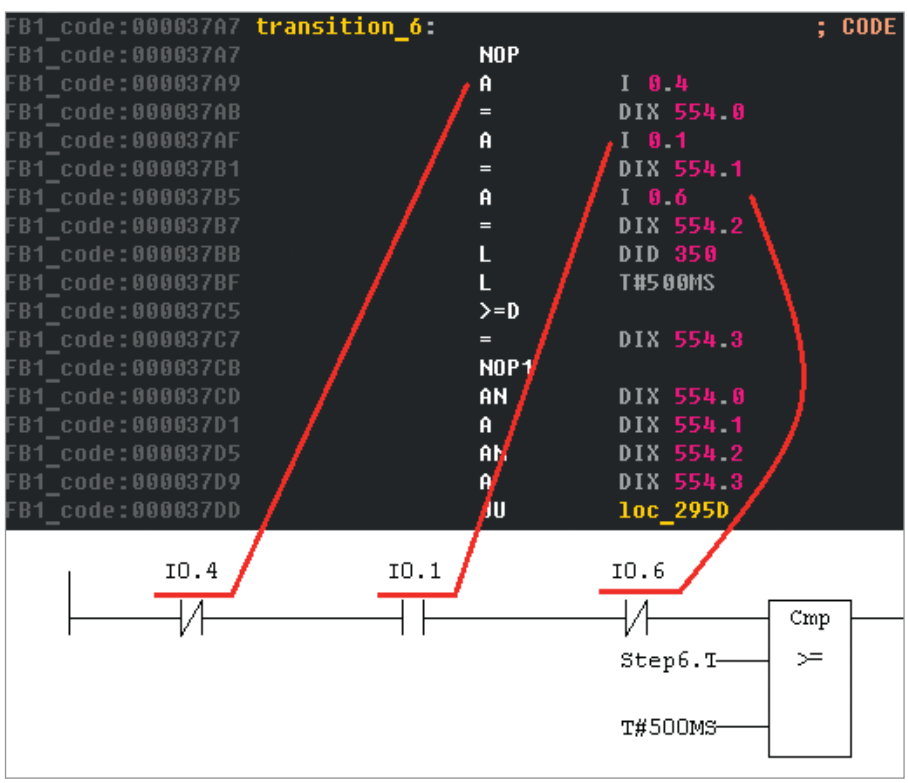

Рис. 5. Соответствие LAD-кода перехода № 6 MC-коду

системы, среды исполнения и сетевые протоколы конфигурирования ПЛК. Также в них используется виртуальная регистровая машина. Её байт-код имеет полуофициальное название MC с добавлением номера версии. В ПЛК SIEMENS S5-101U он именуется MC5, а в SIEMENS S7-300 – MC7. Данный набор команд является практически прямым отображением STL – интерпретации компанией SIEMENS языка IL стандарта МЭК 61131-3. Другие поддерживаемые языки также транслируются в этот байт-код, но по-разному. Для большинства из них возможно однозначное восстановление исходного кода из MC5/7 (ярким исключением является SCL). На рис. 2 изображён участок STL-кода, вызывающий функцию FC6. Он обрамляется фиктивными инструкциями BLD, что позволяет среде (в данном случае – TIA Portal, но механизм одинаков во всех версиях) точно определить границы кода вызова. Выполнив эту задачу, она декодирует передаваемые параметры (рис. 3), расположенные за инструкцией UC вызова блока. Так как виртуальный процессор выполняет выборку команд, строго руководствуясь их последовательным расположением, то компилятору приходится вставлять инструкцию безусловного перехода JU. Она обходит участок с адреса 0x7084 по адрес 0x709f включительно, тем самым пропуская данные параметров. Это позволяет называть архитектуру виртуальной машины фоннеймановской – данные размещаются в той же области памяти, что и код. В конечном счёте среда сможет совершенно точно получить прообраз MC-кода вызова блока AG_RECV на языке FBD.

Несколько слов стоит сказать о языках SIEMENS, не входящих в стандартную поставку программного обеспечения. Наиболее старый из них – GRAPH, имеющий реализации для обеих рассматриваемых серий контроллеров. Его генератор кода обладает рядом интересных особенностей. Результат компиляции даже небольших программ занимает весьма внушительный объём. Например, для тестового примера ZEn02_01_S7GRAPH_Drill финальный размер функционального блока больше 5 кбайт, и это при том, что длина инструкции MC-кода в среднем не превышает 4 байта. Пользовательский код, прикрепляемый к шагам и переходам, выглядит на фоне автоматически сгенерированного очень скромно – всего лишь 6 процентов от тела подпрограммы. Размещается

ПРОГРАММИРУЕМЫЕ ЛОГИЧЕСКИЕ КОНТРОЛЛЕРЫ **REGUL RX00**

для построения ответственных и отказоустойчивых технологических систем

ПЛК REGUL R500

- «горячее» резервирование
- «горячая» замена модулей
- время цикла от 1 мс
- высокоточные измерительные каналы
- веб-интерфейс
- встроенные архивы
- диапазон рабочих температур от –40 до +60°С
- поддержка визуализации
- единое программное обеспечение Epsilon LD с поддержкой 5 языков стандарта IEC 61131-3

ProSoft® SYSTEMS | www.prosoftsystems.ru

он, что характерно, в конце блока. На рис. 4 показан ряд обработчиков, определённых пользователем в терминологии GRAPH, а на рис. 5 – один из переходов в этой же программе.

Впрочем, объём пользовательских программ на языке GRAPH может быть уменьшен за счёт выделения вспомогательного кода в отдельный блок – например в FC72. По умолчанию включена опция компилятора, требующая этого разделения. Из других особенностей результирующего байт-кода стоит выделить часто встречающийся в нём оператор множественного выбора (SWITCH... CASE). Представлен он инструкцией JL, неявно использующей регистр аккумулятора для выбора одного из case-обработчиков. Переходы на них кодируются в инструкциях JU, следующих за JL. Таким образом, в листинге, приведённом на рис. 6, сначала выполнится команда L, затем JL, а после неё – JU loc_52C7.

Ещё один достаточно распространённый язык в контроллерах SIEMENS, недоступный, тем не менее, в базовой версии среды разработки, – SCL. Он был добавлен в STEP 7 и выглядит надстройкой, так как полноценно в неё не интегрируется. Для написания программы на нём необходимо перейти в раздел Sources дерева проекта и создать псевдофайл с исходным текстом на SCL (команда SCL Source из контекстного меню). Редактор, разработанный специально для данного языка, имеет функцию компиляции его кода в MC7-код, создающую на выходе полноценный блок. При попытке его открыть из STEP 7 с ИПП отображается оригинальная программа, доступная для редактирования и последующей сборки в тот же блок. Если же использовать функцию получения проекта Upload Station to PG в изолированном рабочем экземпляре среды разработки (не имеющем исходного проекта), то доступен будет только STL-код, так как средство его декомпиляции в SCL у SIEMENS отсутствует. В отличие от среды CODESYS STEP 7 не умеет передавать исходный текст на ПЛК, поэтому описанный факт имеет место.

Рассмотрев в общих чертах наиболее интересные аспекты работы компиляторов SIEMENS, следует обратить внимание на их отличия в средах STEP 5 и STEP 7. Самых заметных два:

1. Состав блоков. В STEP 5 есть дополнительные кодовые блоки PB и SB, от которых отказались в следующей версии среды.
2. Версии байт-кода. В MC5 присутствуют некоторые инструкции, которых нет в MC7, например:
   - прямой доступ к памяти, в том числе к её областям, которые в S7 будут перемещены в SDB-блоки;
   - прямой доступ к регистрам виртуальной машины (SA, BA, BR, RI, RJ, RS, RT);
   - динамическое создание DB-блоков;
   - динамическая генерация MC5-кода и его выполнение.

Есть также немало различий, непосредственно не затрагивающих результаты работы компиляторов. Это, в первую очередь, форматы исходных проектов. Для STEP 5 была разработана несложная FAT-подобная файловая система, содержащая в себе предопределённые каталоги, объединяющие блоки по их типу. Их существенно больше реально устанавливаемых на устройство – в основном в дополнительных блоках содержатся символьная информация и комментарии.

Устройство среды разработки ПУ ПЛК компании «ПРОСОФТ-Системы»

Epsilon LD – среда разработки, поставляемая компанией «ПРОСОФТ-Системы» для программирования её промышленных контроллеров семейства REGUL, в частности R600. Построена она на базе платформы CODESYS 3, генератор кода настроен на архитектуру микропроцессоров Intel семейства x86. Наибольший интерес в Epsilon LD представляют форматы файлов, так как они необходимы при автоматизации миграции ПУ на поддерживаемые этой средой контроллеры. Главный файл проекта имеет расширение project и является ZIP-архивом с содержимым, примерный вид которого показан на рис. 7.

В целом структура проекта в Epsilon LD испытала на себе сильное влияние платформы Microsoft .NET, так как бóльшая часть программных модулей среды реализована на ней. Основная идея архитектуры подсистемы приложения, обрабатывающей файлы проектов, заключается в следующем. В процессе работы над ними в памяти среды создаются и модифицируются объекты определённых классов .NET. При выполнении же операции сохранения происходит их сериализация, и получившиеся представления сохраняются в отдельных файлах. Далее приведены описания содержимого некоторых из них, представляющих особый интерес:

- __shared_data_storage_string_table__.auxiliary – глобальная таблица строк, которая необходима для уменьшения размера итогового файла. Каждая запись в ней адресуется своим идентификатором и может быть задействована в любом объекте, нуждающемся в хранении текстовой информации, например, строк кода для текстовых языков (IL, ST);
- __shared_data_storage_schema_table__.auxiliary – глобальная таблица описаний классов (схема), экземпляры которых будут создаваться по запросу при открытии проекта. Каждое описание содержит уникальный идентификатор (GUID) класса, порядковый номер и список полей с именами и типами;

```
L     B#16#1
JL    SWITCH(3)
JU    loc_52C9
JU    loc_52C7
JU    loc_52DB
```

Рис. 6. Вид одной из конструкций SWITCH в GRAPH-программе

\private\Codesys3\project\Untitled2.project\

Name	Size
6470a90f-b7cb-43ac-9ae5-94b2338b4573.meta	38
6470a90f-b7cb-43ac-9ae5-94b2338b4573.object	24 072
applicationcontext.e0c003b2-1edd-477a-9148-e4b7c6a4e203.precompileinfo.auxiliary	3 886
c5ebdd22-1f5b-4eb5-b688-77d4af04d61b.meta	42
c5ebdd22-1f5b-4eb5-b688-77d4af04d61b.object	242
logging_project_categories_693fffc3-578c-4dcc-8258-6800165cf1c4.auxiliary	235
poolcontext.pool.00000000-0000-0000-0000-000000000000.precompileinfo.auxiliary	8 392
profile.auxiliary	11 356
profile7.auxiliary	11 356
template.pool.00000000-0000-0000-0000-000000000000.precompileinfo.auxiliary	1 391
__shared_data_storage_schema_table__.auxiliary	171
__shared_data_storage_string_table__.auxiliary	1 632

Рис. 7. Состав тестового проекта

XLight

Серия светодиодных светильников XLD-ДКУ08 для уличного освещения

Применяются для освещения

- скоростных магистралей
- парковок
- пешеходных улиц
- мостов

Универсальная форма КСС позволяет оптимально распределить световой поток для получения максимальной эффективности и равномерности.

Преимущества

- Возможность настройки угла наклона
- Широкий модельный ряд светильников (от 30 до 150 Вт)
- Не требуют обслуживания
- Мгновенное включение
- Снижение нагрузки на сети

IP65 | -40...+50°C | ~220 В | 4200 К | φ > 0,95 | 3 года

EAC

(495) 232-1652 / info@xlight.ru / www.xlight.ru

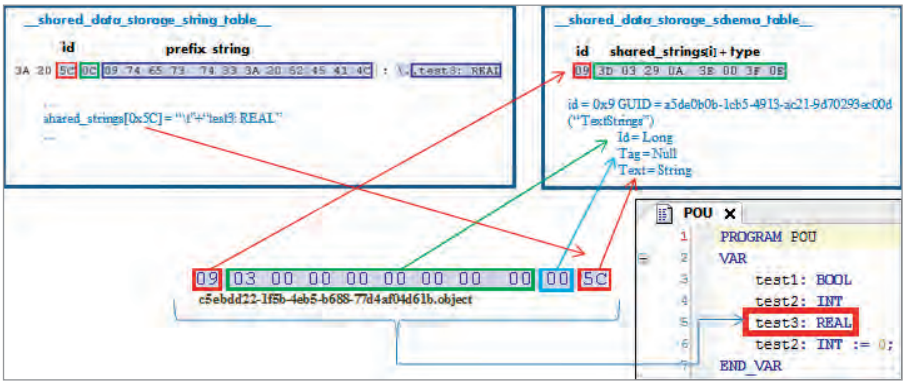

Рис. 8. Разбор содержимого пользовательского объекта POU

- profile7.auxiliary – пакетный файл, состоящий из команд, распознаваемых средой. Основные действия, выполняемые их обработчиками, – создание, заполнение и десериализация объектов. Содержит свою таблицу строк, ссылки на которые могут присутствовать в отдельных полях команд;
- *.object – контейнер, хранящий определённый пользовательский объект. К таковым могут относиться программы (POU), интерфейсы (Interfaces), наборы картинок (Image Pool), настройки проекта (Project Settings), внешние файлы (External Files), библиотечные модули (Libraries). В объектных файлах содержатся экземпляры классов, описанных в схеме, являющихся, по сути, строительными единицами минимального размера. На рис. 8 показано устройство такого экземпляра.

Общий алгоритм обработки проекта при загрузке его средой разработки:

1) открытие архива;
2) обработка файлов с глобальной таблицей строк и схемой;
3) создание пользовательских объектов из файлов объектов;
4) обработка вспомогательных файлов при их наличии;
5) выполнение команд из пакетного файла profile7.auxiliary.

Методы частичной автоматизации процесса миграции ТП

Немалая часть проблем, кроющихся в миграции, была упомянута ранее в том или ином виде, поэтому необходимо подвести итоги и выделить задачи в этом процессе, которые можно решать без привлечения инженера АСУ ТП. Но вначале нужно определить ряд дополнительных терминов. Назовем исходным контроллером (ИК) ПЛК, с которого производится миграция, а целевым контроллером (ЦК) ПЛК, на который она производится. Миграционной парой (МП) обозначим совокупность конкретных ИК и ЦК. В данном разделе под миграцией будем понимать только перенос программ управления ТП, работающих на ПЛК. В табл. 1 перечислен наиболее полный, с точки зрения авторов, список проблем, полученный в результате исследования широкого спектра устройств, а также указана примерная оценка сложности разработки средства автоматизации по десятибалльной шкале.

В первую очередь нужно обозначить части работы, трудно поддающиеся автоматизации, конкретно – проблемы 7 и 8. Они эффективнее решаются с участием эксперта, способного самостоятельно доработать программу управления, – настроить конфигурацию контроллера, дописать недостающий код или заменить некорректный. Все же остальные задачи могут (теоретически) решаться не вручную, а с использованием специального программного обеспечения.

Рассматривая способы автоматизации миграции, стоит начать с закрытия первой проблемы, так как без этого не получится справиться с любой другой из заявленного авторами списка. Задача заключается в том, что имеется набор объектов разных типов, хранящихся в ИПП или ПУ и имеющих отношение к техническому процессу, и их нужно научиться извлекать. Желательно, чтобы в результате проведения этой операции они были восстановлены в первозданном виде. В случае, когда имеются файлы ИПП, она всегда возможна, поскольку её проводит сама среда разработки при открытии ранее сохранённого проекта, но достаточно часто он отсутствует. Так как к ПУ, как правило, предъявляются довольно жёсткие требования – ограничения по размеру и представление кода программы в удобном для исполнения виде, то извлечение объектов из неё может быть произведено не полностью, а с потерей информации. Например, на ПЛК S5-101U при загрузке ПУ не передаётся ряд блоков, в том числе содержащих комментарии и имена входов-выходов. Ещё более неприятным является факт отсутствия на устройстве описания типов переменных, хранящихся в блоках данных (DB, DX). Это вынуждает выводить их из инструкций MC5-кода, что требует полного анализа программы (и вовсе не обязательно будет получена первоначальная структура DB-блока). Для контроллеров, построенных на базе платформы CODESYS, дела обстоят ещё печальнее: в ПУ не содержится какой-либо вспомогательной информации для среды разработки (по умолчанию исходные коды проекта на устройство не загружаются), поэтому однозначно получить объекты не удастся. Несмотря на количество их типов и сложность реализации корректного извлечения каждого из них, технологически наиболее трудоёмким этапом работы является восстановление оригинального кода программы ПЛК. Если среда исполнения представлена виртуальной машиной, то необходим, как минимум, дисассемблер её байт-кода. В случае с SIEMENS его будет достаточно (выдаваемые им мнемоники аналогичны используемым в языке STL), а для реше-

Таблица 1

Список проблем, возникающих при автоматизации процесса миграции

№	Краткая характеристика проблемы	Оценка сложности
1	Различное внутреннее устройство ПУ и ИПП у ПЛК, участвующих в миграции	7
2	Небольшие отличия в синтаксисах одних и тех же МЭК-языков в контроллерах МП	3
3	Отсутствие в ЦК некоторых составных структур данных, доступных в ИК	1
4	Отсутствие в ЦК поддержки МЭК-языка, на котором написана часть исходных программ	5
5	Отсутствие в ЦК некоторых библиотечных функций, доступных в ИК	6
6	Отсутствие в ЦК ряда высокоуровневых абстракций, имеющихся в ИК	7
7	Различия в поддерживаемом контроллерами периферийном оборудовании	9
8	Различия в поддерживаемых контроллерами протоколах взаимодействия с компонентами АСУ ТП	9

ний других производителей потребуется разработать средство декомпиляции исполняемого кода в один из МЭК-языков. Для компиляторных средств CODESYS объём такой работы пропорционален количеству поддерживаемых целевых микропроцессорных архитектур, в машинный код которых может осуществляться трансляция программ.

После разложения исходных двоичных данных на составляющие можно начать выполнять серию преобразований над кодом ПУ. Для этого нужен доступ к промежуточному представлению (IR, Intermediate Representation) программы, формирующемуся во время её восстановления. Оно, хоть и зависит от использовавшегося при разработке языка, будучи основанным на древовидных структурах, достаточно легко обрабатывается и преобразуется в другой вид, более соответствующий языку ЦК. На этих преобразованиях и построена автоматизация проблем 2, 4, 5 и 6. Может показаться, что столь сложный подход излишен и было бы достаточно использования регулярных выражений для модификации текста. Но если простые шаблоны кода и могут ими обнаруживаться, то нетривиальные конструкции требуют обработки на более высоком уровне – их поиск невозможно выполнить посредством детерминированного конечного автомата, состояния которого переключаются входными символами (им фактически и является предварительно скомпилированное регулярное выражение). К тому же код, написанный на графических языках, в любом случае будет представлен в некотором IR. Логично обобщить его обработку на все МЭК-языки, в том числе и текстовые.

В качестве примера рассмотрим более подробно некоторые проблемы, решаемые модификациями IR исходной программы, и сами способы решения.

- Проблема 2: восстановлена исходная программа на языке IL, а его диалект, использующийся в ЦК, отличается незначительными деталями. Решение достаточно тривиально и требует добавления/удаления/модификации узлов древовидной структуры, представляющей IR. В некоторых случаях необходимо добавление новых переменных, например для регистров-аккумуляторов в языке SIEMENS STL.
- Проблема 6: под абстракциями в первую очередь понимаются видные разработчику ПУ особенности архитектуры контроллера, которые отражаются на структуре проекта. Лучший пример – data-блоки SIEMENS, прямых аналогий которым у других производителей, в том числе и у компании «ПРОСОФТ-Системы», не существует. Один из вариантов переноса DB-блока – создание составного типа данных и объявление глобальной переменной этого типа. К сожалению, универсального способа решения нет и в каждом случае нужно находить оптимальное отображение конкретной абстракции.

После извлечения всех необходимых объектов из файлов проекта и выполнения модификаций дерева программы возникает задача объединить все имеющиеся данные, записав их в ИПП среды разработки целевого ПЛК. Она решается в контексте проблемы 1 заявленного списка проблем. Обычно этап упаковки не составляет особых трудностей – здесь нет никакой дополнительной потери информации, поэтому детально он рассматриваться не будет.

Заключение

В целом задача автоматизации миграции технологического процесса между программно-аппаратными комплексами различных производителей является достаточно сложной. Разнообразие устройств и программного обеспечения в области АСУ ТП существенно повышает требования к функциональности программного средства, оптимизирующего перенос имеющейся инфраструктуры на новое оборудование. Конечно, ряд этапов миграционного процесса требует непосредственного участия человека, так как представление общей архитектуры системы, которое будет учитываться при выборе альтернативы компоненту, подвергающемуся миграции, сложно формализовать и описать понятным для ЭВМ языком. Но есть и достаточно сложные подзадачи, неплохо (иногда практически полностью) поддающиеся автоматизации. В основном они связаны с преобразованиями исходного кода программы управления.

Подводя итоги, можно с уверенностью утверждать, что, обладая пониманием внутреннего устройства промышленных контроллеров и инструментального ПО, можно создать программный комплекс, существенно ускоряющий миграцию ПУ. ●

E-mail: surgeon@ntcsiz.ru