FASTWEL I/O: развитие продуктовой линейки

Часть 1

Александр Локотков

В статье описана эволюция программируемых контроллеров серии FASTWEL I/O с момента выхода цикла публикаций «FASTWEL I/O изнутри» в 2007–2008 годах до настоящего времени. Даётся подробное описание некоторых функциональных возможностей, отличающих FASTWEL I/O от контроллеров других производителей.

Введение

Прошло семь лет с момента выхода цикла публикаций «FASTWEL I/O изнутри», в которых с позиции разработчика рассказывалось о функциональных возможностях, особенностях и принципах, положенных в основу разработки аппаратно-программного комплекса FASTWEL I/O.

За это время состав продуктовой линейки FASTWEL I/O пополнился новыми контроллерами и модулями ввода-вывода, расширились функциональные возможности ранее разработанных модулей, а также инструментального и системного программного обеспечения. Кроме того, реализована поддержка дополнительных промышленных сетевых протоколов и расширены коммуникационные возможности контроллеров. При этом решающее влияние на развитие аппаратно-программного комплекса FASTWEL I/O оказано пользователями, реальными и потенциальными, включая разработчиков систем промышленной автоматизации и бортовых систем, а также заинтересованных специалистов, высказавших массу интересных замечаний и пожеланий по следам первых публикаций в «СТА» 1–4/2007 и 1/2008.

Цель данной статьи – продемонстрировать эволюцию FASTWEL I/O широкой аудитории существующих и потенциальных пользователей, отметить отличительные особенности комплекса в сравнении с вариантом семилетней давности и более развёрнуто ответить на ряд вопросов концептуального характера, задаваемых пользователями.

Для краткости изложения термин «контроллер» будет далее использоваться вместо термина «контроллер узла сети», которым в документации и справочных материалах обозначается модуль центрального процессора, исполняющий приложение пользователя, разработанное в адаптированной среде разработки CoDeSys 2.3, и взаимодействующий с объектом автоматизации через присоединённые к нему модули ввода-вывода.

Эволюция FASTWEL I/O в кратком изложении

На момент выхода цикла статей «FASTWEL I/O изнутри» в 2007–2008 годах линейка FASTWEL I/O включала в себя три контроллера: CPM701, CPM702, CPM703 на базе 16-разрядного микропроцессора R1610C, совместимого с 80186, а также базового набора модулей дискретного ввода-вывода, аналогового ввода, вспомогательных модулей ввода и распределения потенциалов питания датчиков и коммутационных устройств и модулей ввода питания межмодульной шины.

Контроллеры обладали довольно ограниченными вычислительными ресурсами: размер памяти для размещения кода пользовательского приложения составлял не более 64 кбайт, память переменных занимала не более 32 кбайт, на области ввода-вывода приходилось по 8 кбайт. При этом в составе контроллеров не было часов-календаря с питанием от батареи и на системном уровне не поддерживались энергонезависимые (RETAIN) переменные, а коммуникационные возможности контроллеров были представлены сервисами подчинённого узла сетевых протоколов CANopen (CPM701), MODBUS RTU/ASCII (CPM702) и MODBUS TCP (CPM703).

Встроенная система исполнения приложений, создаваемых пользователем в среде разработки CoDeSys 2.3 на языках стандарта МЭК 61131-3, из-за ряда ограничений CoDeSys 2.3, касающихся целевой платформы 80186, не поддерживала многозадачный режим выполнения прикладных алгоритмов. А вследствие довольно небольшого размера оперативной памяти контроллеров было невозможно использовать механизм обновления выполняющегося приложения без его остановки и перезапуска контроллера.

Сервис обмена данными с модулями ввода-вывода по внутренней шине FBUS поддерживал единственный режим работы, при использовании которого на каждом цикле шины контроллер передавал модулям один групповой запрос, содержащий данные для всех выходных каналов системы, после чего получал от модулей групповой ответ с данными от всех входных каналов. Во время загрузки или обновления приложения в контроллере из среды разработки CoDeSys 2.3 информационный обмен с модулями ввода-вывода прекращался, поскольку из-за малой вычислительной мощности не удавалось обеспечить устойчивое соединение

контроллера со средой разработки CoDeSys 2.3 во время загрузки приложения при одновременном интенсивном обмене по внутренней шине.

Перечисленные особенности и возможность функционирования в широком диапазоне температур ограничивали область применения FASTWEL I/O бортовыми системами для железнодорожного транспорта, небольшими системами сбора данных и управления дискретными и, с некоторыми допущениями, периодическими технологическими процессами. Однако реальные и потенциальные пользователи, заинтересованные в расширении возможностей применения FASTWEL I/O, заставляли нас двигаться вперёд, преодолевая первоначальные ограничения и добавляя новые функциональные возможности.

Итогом работы инженеров компании стало начало производства в 2009 году более десятка новых типов модулей ввода-вывода, включая модули измерения температуры (AIM724, AIM725), аналогового вывода (AIM730, AIM731), многофункциональный модуль дискретного ввода (DIM764), модули аналогового ввода сигналов постоянного тока 0–20 мА и 4–20 мА (AIM721, AIM722, AIM723), последовательных интерфейсов RS-485 и RS-232C (NIM741, NIM742) и контроллер CPM704 с сетевым интерфейсом подчинённого узла PROFIBUS DP-V1.

В то же время для контроллеров CPM70x была разработана многозадачная система исполнения приложений CoDeSys 2.3, поддерживающая обновление приложений на лету, а для обмена с модулями ввода-вывода был добавлен режим индивидуального опроса, при котором кратковременное или постоянное отсутствие связи с каким-либо модулем не приводило к потере связи со всеми модулями.

К концу 2011 года стали доступными три новых контроллера (CPM711, CPM712 и CPM713) на базе 32-разрядного x86-совместимого процессора Vortex86DX с тактовой частотой 600 МГц, имеющих на системном уровне поддержку энергонезависимых переменных, встроенные часы-календарь с батарейным питанием, в среднем в 20 раз более высокое быстродействие и в 30 раз бóльшие размеры памяти для размещения кода и данных приложения, чем у ранее разработанных CPM70x, но с сохранением потребляемой мощности и с возможностью миграции проектов CoDeSys 2.3, ранее разработанных для CPM701, CPM702 и CPM703, на CPM711, CPM712 и CPM713 соответственно.

Коммуникационные возможности контроллера CPM713 по сравнению с CPM703 были расширены функционирующим одновременно с сервисом подчинённого узла мастером протокола MODBUS TCP и системной библиотекой сокетов FastwelSysLibSockets.lib, поставляемой в пакете адаптации CoDeSys 2.3 для FASTWEL I/O и позволяющей реализовать любые сетевые протоколы поверх UDP и TCP в приложении CoDeSys 2.3. Контроллер CPM712 вышел с поддержкой функций мастера протокола MODBUS RTU/ASCII, а информационная ёмкость сервиса протокола CANopen контроллера CPM711 по сравнению с CPM701 была увеличена более чем в 3 раза и достигла 512 передаваемых и 512 принимаемых коммуникационных объектов.

С появлением контроллеров, способных обрабатывать большие объёмы данных, потребовалось увеличить информационную ёмкость FASTWEL I/O в части количества вводимых аналоговых сигналов и дополнительных коммуникационных интерфейсов.

Кроме того, для реализации некоторых классов систем сбора данных и управления пользователям требовалась возможность обнаружения отказов измерительных и дискретных каналов, включая обрыв цепи присоединения датчика и входного канала модуля ввода-вывода.

В итоге в течение 2013 года разработаны 8-канальные модули дискретного ввода с контролем целостности цепей присоединения датчиков и 8-канальные многодиапазонные модули аналогового ввода с расширенной диагностикой и суммарным временем измерения по всем каналам чуть более 1 мс, а в серийно выпускаемый модуль приёма сигналов термометров сопротивления AIM725 добавлена функция обнаружения обрыва и короткого замыкания измерительных цепей. Одновременно с указанными разработками было выпущено специальное исполнение модуля AIM725, поддерживающее номинальные статические характеристики термометров сопротивления отечественного производства ТСП 50П, ТСП 100П, ТСМ 50М и ТСМ 100М, а в серийно выпускаемом модуле ввода сигналов термопар AIM724 реализована поддержка термопар типа L (XK).

Коммуникационные возможности всех контроллеров FASTWEL I/O в июле 2013 года существенно расширились с выходом встроенного в систему исполнения драйвера коммуникационных портов RS-485 и RS-232C на основе модулей NIM741 и NIM742, подключаемых к межмодульной шине контроллеров. Ранее для организации дополнительных каналов обмена данными через указанные модули в приложении, выполняющемся на контроллере, требовалось использовать функциональные блоки из библиотеки nim741_742.lib, имеющие не очень простую для понимания модель поведения и занимающие приличное количество памяти. Теперь же для приёма и передачи данных по последовательным каналам связи через модули NIM741 и NIM742 можно использовать две простые функции из библиотеки FastwelSysLibCom.lib.

Начало 2014 года было отмечено выпуском новой версии (2.62) системного программного обеспечения контроллеров и пакета адаптации CoDeSys 2.3 для FASTWEL I/O, в которой для контроллеров CPM711, CPM712 и CPM713 появилась поддержка интеграции с GPS-приёмником для определения точного времени и синхронизации встроенных системных часов контроллера.

Кроме того, контроллер CPM713 стал поддерживать протокол NTP для синхронизации времени по сети, причём как в качестве клиента, так и сервера сетевого времени, что в совокупности с возможностью получения точного времени от GPS-приёмника позволило решить проблему обеспечения единства времени на множестве узлов сети.

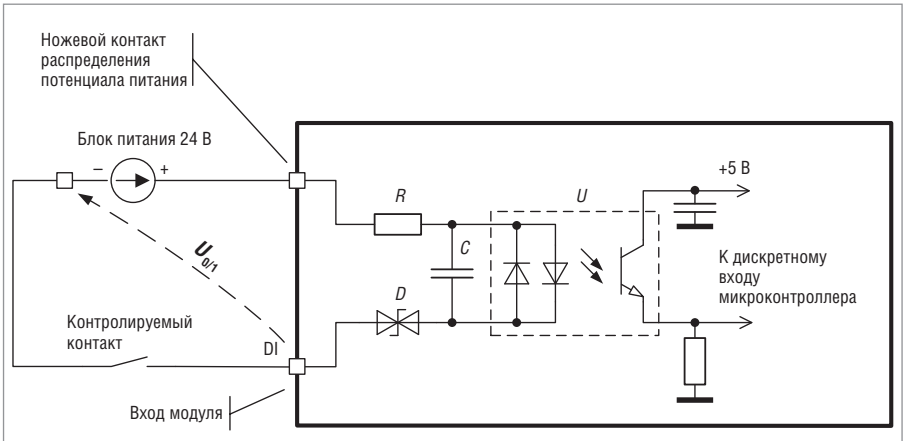
И, наконец, в контроллерах CPM712 и CPM713 был реализован сервис подчинённого узла сетевого протокола DNP3 Outstation с уровнем совместимости, превышающим Level 2.

Далее будет рассказано о некоторых из перечисленных новшеств чуть более подробно.

Контроль целостности цепей подключения датчиков

В некоторых отраслях энергетики к системам автоматизации предъявляются весьма жёсткие требования, касающиеся диагностирования исправного состояния технических средств системы.

Например, в документе «Основные положения по автоматизации, телеме-



Условные обозначения: ***R*** – резистор; ***C*** – конденсатор; ***U*** – напряжение (***U*$_0$** – канал выключен, ***U*$_1$** – канал включён); ***D*** – стабилитрон; **DI** – дискретный вход.

Рис. 1. Упрощённая схема входной цепи модуля дискретного ввода DIM762

ханизации и информационно-управляющим системам газоперерабатывающих производств», выпущенном в 1997 году (тогда еще РАО «Газпром»), указывалось, что любые отключения каналов контроля параметров, определяющих взрывоопасность объекта или изменение параметров системы защиты, должны фиксироваться системой.

Аналогичный нормативный документ СО 01-05-АКТНП-002-2004, выпущенный ГУП «ИПТЭР» с участием ОАО «АК «Транснефтепродукт», среди требований к контроллерам для систем автоматизации перекачивающих станций содержал следующие положения, касающиеся функций диагностики: «Аппаратные устройства контроллеров должны иметь средства самоконтроля, обеспечивающие их тестирование:

- функционирования активных элементов;
- программ пользователя;
- интерфейсных каналов и цепей датчиков;
- функционирования модулей ввода-вывода.

При обнаружении неисправности должны определяться характер и место неисправности, формироваться сигналы, которые могут быть использованы для принятия мер по устранению последствий отказа».

Первоначально при разработке модулей аналогового и дискретного ввода FASTWEL I/O функции контроля целостности цепей подключения датчиков не было уделено должного внимания ввиду сделанного тогда предположения, что потребители будут не готовы за неё платить, а также из-за существенных ограничений по габаритам. Однако для некоторых системных интеграторов отсутствие контроля целостности цепей оказалось существенным фактором, ограничивающим спектр применений FASTWEL I/O.

Быстро удалось реализовать функцию обнаружения короткого замыкания и обрыва цепей подключения термометров сопротивления для модуля AIM725, поскольку это потребовало только модификации микропрограммы модуля без доработки аппаратной части. По-

путно разработчики смогли уменьшить время измерения с первоначальных 200 мс на один канал до 80 мс и ввести ещё ряд улучшений, одно из которых привело к появлению нового исполнения модуля, предназначенного для применения совместно с термометрами сопротивления, имеющими номинальные статические характеристики ТСП и ТСМ по ГОСТ 6651-2009.

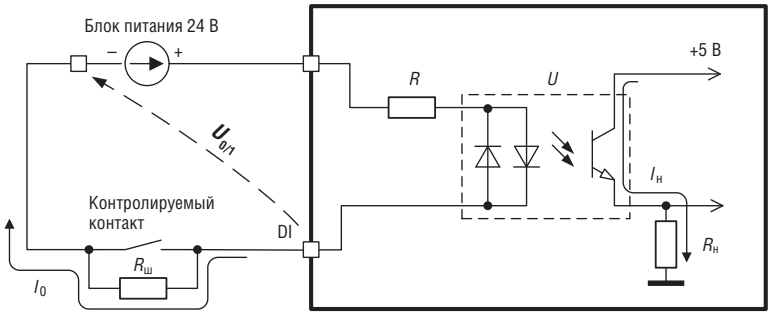
Для реализации функции обнаружения обрыва термопары в модуле AIM724 потребовалась небольшая доработка аппаратной части и модификация микропрограммы модуля. Измерительный тракт модуля содержит схему обнаружения обрыва термопары (Burnout Detection Circuit), которая изначально не была задействована, а после модификации периодически формирует ток сверхмалой величины, пропускаемый через спай подключённой термопары сначала в одном, а затем в противоположном направлении, что по полученной разности потенциалов на входном инструментальном усилителе позволяет судить о целостности спая и цепи подключения термопары ко входу модуля.

В новых 8-канальных модулях AIM791 и AIM792, предназначенных для измерения тока и напряжения, имеется возможность установки допустимых границ диапазонов измерения для каждого канала, выход за пределы которых сопровождается установкой соответствующих признаков в статусном канале модуля, а также светодиодной индикацией. Кроме того, при обрыве цепи любого датчика с токовым выходом, подключённого к каналам модуля AIM791, в статусном канале модуля устанавливается признак обрыва цепи.

Наиболее сложной для решения оказалась проблема контроля целостности цепи датчиков дискретных сигналов, и прежде всего так называемых «сухих» контактов, к которым относятся механические контакты концевых выключателей, блок-контакты магнитных пускателей, контакторов, промежуточных реле и других коммутационных приборов. Сложность состоит в том, что модулю требуется отличать разомкнутое состояние контролируемого контакта от обрыва цепи подключения контакта к входному каналу модуля, то есть вместо двух логических состояний каждый канал модуля должен различать три: включён, выключен или обрыв цепи.

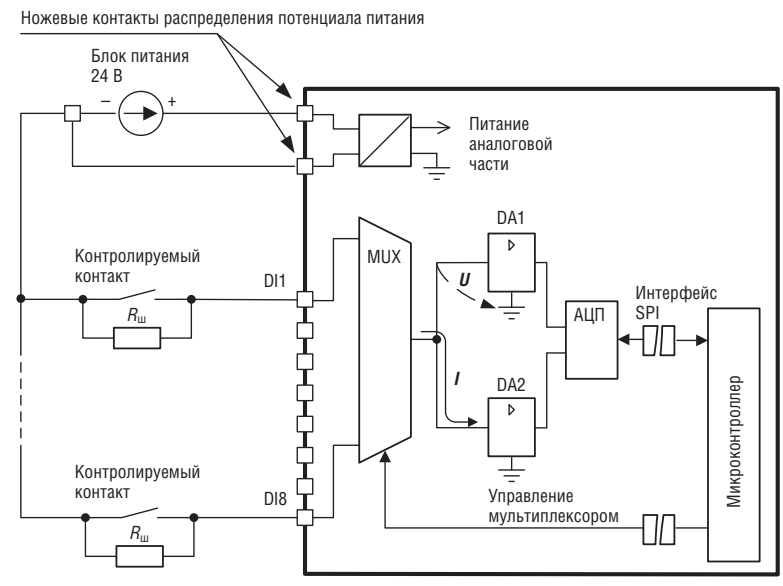
Описание способа решения данной задачи на примере модуля DIM766 следует начать с анализа упрощённой схемы канала модуля дискретного ввода DIM762 (рис. 1), применяемой многими производителями многоканальных модулей дискретного ввода с однопроводным подключением, в которых не предусмотрено возможности определения состояния обрыва цепи.

Каждый из восьми каналов модуля DIM762 позволяет определить два состояния подключённого к нему датчика (контакта или дискретного выхода с ненулевым током утечки): включён или выключен. Положительный потенциал +24 В с выхода блока питания подключается к резистору *R* во входной цепи каждого из восьми каналов через ножевой контакт распределения питания. Общий провод выхода блока питания, подключаемый ко второму ножевому контакту, на рис. 1 не показан, так как это не требуется для последующего изложения. Резистор *R* ограничивает ток во входной цепи. Значение тока ограничения составляет 10 мА. Совместно с конденсатором *C* резистор *R* определяет постоянную времени фильтра нижних частот, устраняющего высокочастотные помехи во входной цепи. Стабилитрон *D* определяет диапазон значений напряжения U_0, при которых канал нахо-



Условные обозначения: ***R*** – резистор (***R*ш** – шунтирующий резистор); ***U*** – напряжение (***U*0** – канал выключен, ***U*1** – канал включён); ***I*0** – ток во входной цепи при разомкнутом контакте; **DI** – дискретный вход.

Рис. 2. Схема с шунтирующим резистором $R_ш$, подключённым параллельно контролируемому контакту



Условные обозначения: ***R*ш** – шунтирующий резистор); ***U*** – напряжение; **DI** – дискретный вход; **MUX** – мультиплексор; **DA1** – узел измерения напряжения; **DA2** – узел измерения тока.

Рис. 3. Упрощённая схема входных цепей модуля дискретного ввода DIM766

дится в выключенном состоянии (от +15 до +30 В), и U_1, при которых канал включён (от –3 до +5 В).

При обрыве цепи подключения контролируемого контакта ко входу модуля разность потенциалов между входом DI и общим проводом блока питания 24 В будет равна U_0, то есть будет соответствовать состоянию, когда контролируемый контакт выключен. Таким образом, для того чтобы различить эти два состояния, необходимо при помощи резистора, включённого параллельно контакту, обеспечить протекание некоторого различимого тока по входной цепи, когда контролируемый контакт разомкнут (рис. 2), и сделать так, чтобы оптрон *U* из логического элемента с двумя состояниями превратился, как минимум, в двухразрядный АЦП.

Для доступных на рынке блок-контактов, концевых выключателей и реле обычно указывается минимальное значение тока, протекание которого через замкнутый контакт гарантируется производителем при некотором номинальном напряжении, приложенном к контакту в замкнутом состоянии, скажем, 10 мА при 30 В.

Для уверенного распознавания разомкнутого состояния контакта в полном диапазоне рабочих температур и значений выходного напряжения источника питания ток через шунтирующий резистор $R_ш$ должен быть много меньше минимального тока во включённом состоянии и много больше максимального тока, протекающего по входной цепи модуля при обрыве цепи подключения контролируемого контакта ко входу модуля. В электронике понятие «много меньше» или «много больше» обычно выражается в количественном отношении в 5–10 раз. Исходя из этого, получаем следующие оценки пороговых значений тока и напряжения для входной цепи модуля, упрощённая схема которой показана на рис. 2:

1. Уровень логической единицы:
   0 В ≤ U_1 ≤ 5 В при I_1 ≥ 10 мА.
2. Уровень логического нуля:
   15 В ≤ U_0 ≤ 30 В при 1 мА ≤ I_0 ≤ 2 мА.
3. Обрыв цепи: от 15 В ≤ U_0 ≤ 30 В
   при 0 мкА ≤ I_0 < 200 мкА.

Таким образом, для распознавания уровня логической единицы через контролируемый контакт в замкнутом состоянии должен протекать ток существенной величины, что означает высокое энергопотребление системы с большим количеством дискретных входов.

Даже если предположить, что все контролируемые контакты бóльшую часть времени разомкнуты, ток величиной 2 мА, протекающий через каждый шунтирующий резистор $R_ш$, для контроллера с 64 дискретными входами приводит к увеличению потребляемой мощности на 3 Вт.

Решить данную проблему можно только путём использования импульсного режима оценки состояния каждого канала, когда ток пропускается через контролируемую цепь на короткий промежуток времени, в течение которого проводится оценка текущего логического состояния.

В этой связи при проектировании модуля DIM766 классическая схема входных цепей модуля дискретного ввода, показанная на рис. 1, была преобразована в схему многоканального модуля аналогового ввода, позволяющего оценивать сопротивление подключённых к его входам цепей, как показано на рис. 3.

Микроконтроллер через цепи опторазвязки поочерёдно подключает каждый вход модуля к выходу мультиплексора MUX, который связан с узлом измерения напряжения DA1 и узлом измерения тока DA2. Очевидно, что при этом остальные семь входов модуля отключены от измерительных цепей и по ним не протекает ток от источника питания 24 В.

Выходы узла измерения напряжения DA1 и узла измерения тока DA2 под-

ключены к двум входам аналого-цифрового преобразователя (АЦП), которые опрашиваются микроконтроллером один за другим. Исходя из полученных результатов, можно судить о состоянии контролируемой цепи, подключённой ко входу модуля. Далее микроконтроллер повторяет описанную последовательность действий для следующего входа и так далее.

Суммарное время опроса всех восьми каналов модуля составляет не более 1,4 мс, включая контроль наличия напряжения питания датчиков, поданного на ножевые контакты распределения питания. Для модуля установлены уровни определения логических состояний контролируемых цепей в зависимости от режима канала в соответствии с таблицей 1.

При конфигурировании модуля могут быть заданы режимы работы каждого канала, значения сопротивления шунтирующих резисторов из ряда от 1,8 до 33,0 кОм, а также программная задержка определения состояния каждого канала для фильтрации дребезга контактов.

Приложению CoDeSys 2.3 доступны логические состояния каждого канала, признаки обрыва цепей подключения датчиков, диагностическая информация о состоянии измерительного тракта и о наличии напряжения питания датчиков, поданного на ножевые контакты модуля. Кроме того, приложение может использовать результаты измерения тока и напряжения в цепи каждого канала, что позволяет пользовате-

Таблица 1

Параметры определения логических состояний входов модуля DIM766

Логическое состояние	Режим цифрового входа по ГОСТ Р 51841-2001	
	Тип 1	Тип 2
«1» (включён)	0,0…5,0 В при токе более 250 мкА	Ток 2,0…15,0 мА
«0» (выключен)	Более 16,1 В	Ток 0,25…1,50 мА
Обрыв цепи	При токе менее 200 мкА	При токе менее 200 мкА

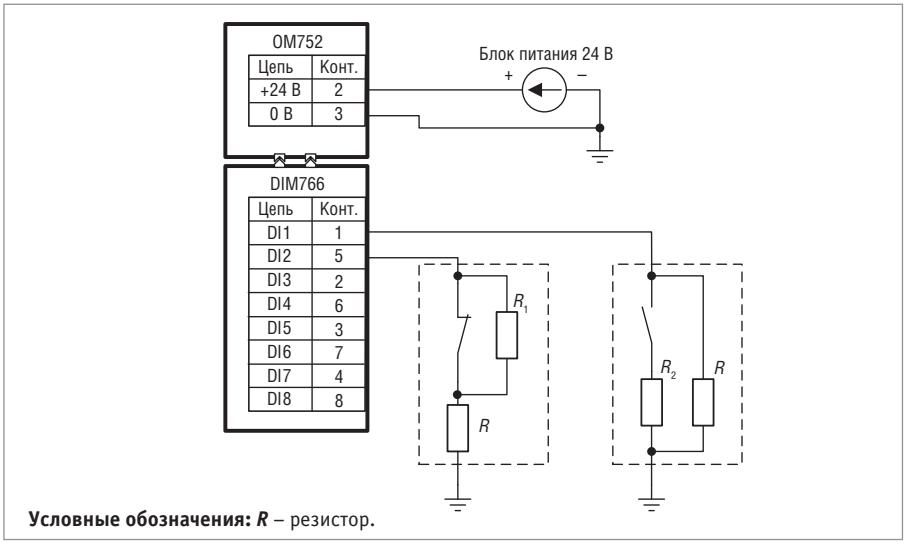
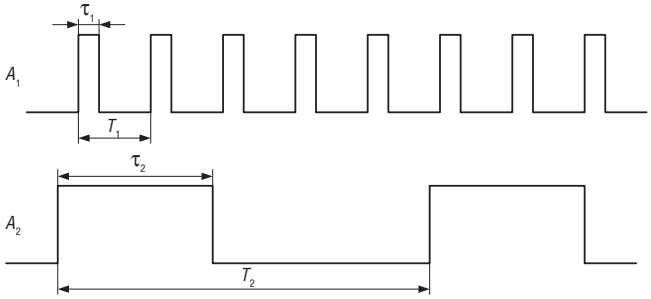


Рис. 4. Схема подключения неадресных нормально-замкнутых и нормально-разомкнутых извещателей к каналам модуля DIM766

лю самостоятельно оценивать параметры контролируемых цепей. Например, можно использовать DIM766 для подключения неадресных извещателей пожарной и охранной сигнализации, как показано на рис. 4. При номиналах: R_1 = 20 кОм, R_2 = 4,7 кОм, R = =10 кОм и напряжении питания 24 В, следует в конфигурации модуля для всех каналов установить задержку включения 300 мс, а параметры контроля цепи *Тип 1: шунт 10 кОм*. Состояние обрыва определяется модулем самостоятельно, а состояние короткого замыкания и рабочие состояния извещателей следует определять в приложении CoDeSys 2.3, вычисляя отношение значения напряжения на входе модуля к напряжению питания. Для состояний «КЗ», «Тревога НРЗ», «Норма» и «Тревога НЗ» данные отношения составят 0,00...0,40; 0,40...0,70; 0,70...0,87 и 0,87...0,95 соответственно.



1. A_1 более срочный по сравнению с A_2, если $T_1 << T_2$
2. A_1 более быстрый по сравнению с A_2, если $\tau_1 << \tau_2$

Условные обозначения: ***A*** – алгоритм; ***T*** – период цикла; **τ** – время выполнения.

Рис. 5. Описание и циклограммы алгоритмов разной срочности и длительности выполнения

Модуль имеет в своём составе восемь светодиодных индикаторов, отображающих текущее состояние каждого канала, включая обрыв цепи, для чего применяется специальная последовательность включения и выключения светодиода.

Последней отличительной особенностью DIM766 является наличие средств защиты входных цепей модуля от воздействия микросекундных и наносекундных помех большой энергии.

Развитие средств разработки и исполнения приложений

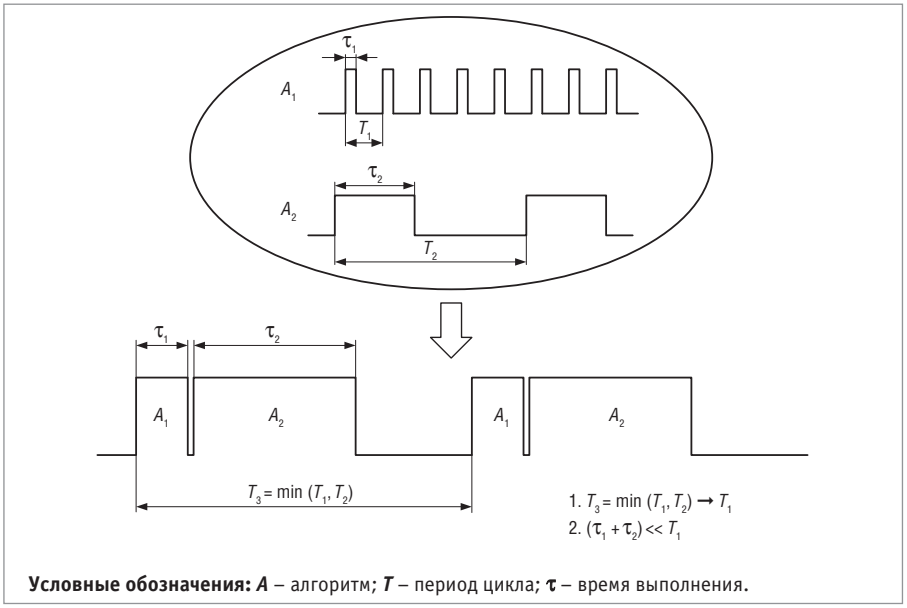
Со времён первого в истории программируемого логического контроллера (Modicon, модель 084, 1969 г.) принцип работы большинства современных программируемых контроллеров остался практически неизменным. Пользовательский алгоритм, загруженный в контроллер в виде набора инструкций встроенного процессора или интерпретатора, выполняется циклически, перед каждым циклом считывая значения входных сигналов и формируя выходные сигналы. То есть по существу процессор контроллера многократно вызывает одну и ту же последовательность инструкций пользовательского алгоритма, предваряя каждый вызов обновлением переменных, связанных с входными каналами периферийных модулей контроллера и поступающих по сети, а после вызова формирует новые значения и логические состояния для

выходных каналов модулей и сообщений, передаваемых по сети.

Данная вычислительная модель весьма проста для понимания и вполне пригодна для решения многих задач сбора данных и управления. Однако при необходимости выполнения нескольких алгоритмов на одном контроллере довольно часто оказывается, что некоторые из них должны выполняться чаще других, более длительных алгоритмов. Например, алгоритм регулирования температуры может требовать большого количества вычислений, в том числе с плавающей точкой, с периодом от сотен миллисекунд до единиц секунд, тогда как фильтрация дребезга контактов при первичной обработке состояний каналов дискретного ввода состоит из сравнительно небольшого количества операций, вызываемых с периодом от десятков и даже единиц миллисекунд. Данный пример иллюстрирует рис. 5.

Далее будем называть алгоритмы, требующие большого количества вычислений с небольшой частотой, длительными и несрочными, а их антагонистов – быстрыми и срочными.

Обратите внимание, что срочность является сравнительной характеристикой для нескольких алгоритмов или вычислительных процессов и позволяет на качественном уровне понять, что период выполнения одного из них много меньше периода второго. Если же время выполнения одного алгоритма много меньше остальных, то он более быстрый. Соответственно, оставшиеся алгоритмы являются более длительными.



Условные обозначения: ***A*** – алгоритм; ***T*** – период цикла; **τ** – время выполнения.

Рис. 6. Циклограммы исполнения алгоритмов разной срочности и длительности на однозадачном контроллере

Если вызывать быстрый алгоритм вслед за длительным на одном процессоре, то делать это нужно с периодом быстрого алгоритма, как показано на рис. 6. При этом производительность процессора должна быть достаточной, чтобы успеть выполнить все операции быстрого и длительного алгоритмов в пределах требуемого периода цикла быстрого.

С другой стороны, время измерения параметров процесса, управляемого длительным алгоритмом, во многих случаях соизмеримо с периодом данного алгоритма, а это значит, что длительный алгоритм, вызываемый с периодом быстрого, в большей части своих циклов будет оперировать неизменившимися входными данными, если только в нём не предусмотрены средства слежения за их актуальностью.

Самый простой способ решения обозначенной проблемы состоит в применении контроллера с достаточно быстрым процессором, способным выполнить все прикладные алгоритмы с использованием кратчайшего требуемого периода цикла.

Однако далеко не каждый, даже опытный инженер в состоянии точно оценить, какая именно производительность будет достаточной для реализации требуемых функциональных возможностей системы, особенно если нужно учитывать последующую возможность расширения.

Кроме того, использование мощных процессоров для широкого спектра задач автоматизации не всегда оправданно с точки зрения затрат на аппаратные средства системы, а в ряде случаев и невозможно, скажем, из-за чрезмерного энергопотребления, размеров или длительного времени запуска контроллера при включении питания.

Для решения данной проблемы в стандарте МЭК 61131-3 было определено понятие *задача* (task), олицетворяющее реальный или виртуальный процессор, способный выполнять связанную с ним последовательность программных единиц (Program Organization Units, или POU) при наступлении заданного условия запуска. Таким образом, задача является элементом управления программными единицами, определяя для них контекст выполнения, который состоит, как минимум, из указателя текущей выполняемой инструкции, указателя стека и набора регистров реального процессора или интерпретатора.

Программа, выполняемая в контексте циклической задачи, вызывается с периодом, заданным для этой задачи, а для программы, выполняемой в контексте ациклической задачи, условием запуска является передний фронт некоторой булевой переменной.

Для задачи в стандарте введено понятие *приоритета*, отражающее срочность выполняемых в её контексте программных единиц по сравнению с программными единицами, выполняемыми в контексте других задач. Приоритеты задач определяют формальный критерий, согласно которому система исполнения контроллера предоставляет физический процессор той или иной задаче: если в некоторый момент времени должны выполняться или уже запущены несколько задач, то доступ к процессору получает задача с наивысшим приоритетом. Процесс определения задачи, которой в некоторый момент времени должен быть отдан процессор (или одно из ядер многоядерного процессора), называется *планированием задач* (tasks scheduling), а компонент системного программного обеспечения, занимающийся планированием задач, называется *планировщиком* (scheduler). Следует отметить, что согласно МЭК 61131-3 планирование может осуществляться как с вытеснением менее приоритетных задач (preemptive scheduling), так и без вытеснения (non-preemptive).

Программными единицами в МЭК 61131-3 второй редакции, выпущенной в 2003 году, могут быть *программы* (PROGRAM), *функциональные блоки* (FUNCTIONAL BLOCK) и *функции* (FUNCTION), которые описываются наборами входных, выходных и внутренних переменных и содержат исполняемую при вызове последовательность инструкций. Перед вызовом программы или блока входные переменные вводятся системой исполнения контроллера из других программ, функциональных блоков или из окружения, каковым является сетевая подсистема контроллера и подсистема ввода-вывода, а выходные переменные выводятся другим программам, блокам или в окружение по окончании очередного исполнения. Внутренние переменные программ и функциональных блоков в отличие от внутренних переменных функций, сохраняют свои значения между вызовами.

Принципиальное отличие между программой и функциональным блоком состоит в том, что в приложении контроллера может быть только один экземпляр программы с некоторым именем, а экземпляры функционального блока можно объявлять в виде переменных среди внутренних переменных программ и вызывать из их кода.

Функция (FUNCTION) очень похожа на программу (PROGRAM), но не имеет внутренних переменных, значения которых сохраняются между вызовами, то есть время жизни функции ограничивается временем между началом её выполнения в точке вызова и возвратом на следующую инструкцию после точки вызова, тогда как время жизни программ и функциональных блоков ограничено временем функционирования контроллера. При использовании энергонезависимых переменных внутри функциональных блоков и программ время жизни программ и функциональных блоков теоретически не ограничено.

Планирование без вытеснения (non-preemptive) означает, что если одна из задач получила процессор, то до завершения её текущего цикла процессор не будет предоставлен ни одной другой задаче. При этом в случае если в момент принятия решения о предоставлении процессора сразу несколько задач готовы к выполнению и имеют одинаковые приоритеты, то процессор предоставляется задаче, которая до этого момента ожидала его дольше всех задач равного с ней приоритета.

Планирование с вытеснением (preemptive) означает, что если во время выполнения некоторой задачи возникли условия для выполнения другой, более приоритетной задачи, то текущая задача приостанавливается, а процессор немедленно предоставляется более приоритетной задаче.

Планирование без вытеснения, применительно к проблеме координации выполнения нескольких задач разной срочности и длительности, не добавляет ровным счётом никаких новых возможностей к классической однозадачной системе исполнения контроллера – менее срочные и более длительные задачи, получив процессор, будут откладывать запуск более быстрых и срочных задач до своего завершения.

Планирование с вытеснением, напротив, позволяет решить проблему, если быстрым и срочным задачам назначен более высокий приоритет, чем длительным и менее срочным, как показано на рис. 7.

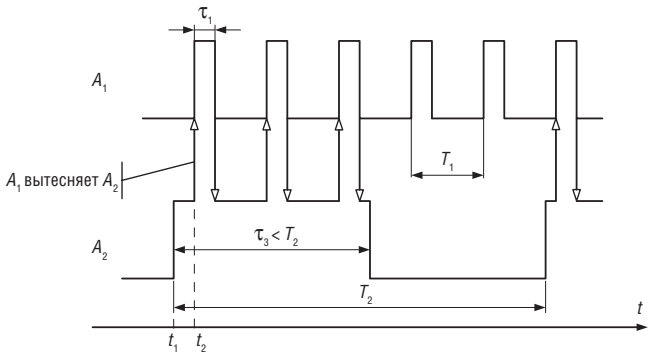
В момент t_1 процессор «отдаётся» задаче A_2, которая выполняется до момента t_2, при наступлении которого A_2 приостанавливается и ожидает завершения очередного цикла более приоритетной задачи A_1, после чего A_2 опять «получает» процессор. Обратите внимание, что несмотря на трёхкратное вытеснение A_2, время τ_3, требуемое для выполнения очередного цикла A_2, в данном случае не превышает длительности периода T_2, заданного для A_2. Та-

ким образом, более быстрая и срочная задача A_1 всегда «получает» процессор в требуемые моменты времени и процессор используется только тогда, когда это действительно нужно для правильной работы приложения.

Циклограмма на рис. 7 иллюстрирует идеальный случай, когда все вычислительные процессы контроллера уложены в две циклические задачи, а ввод-вывод и передача процессора от одной задачи к другой, называемая переключением контекста, происходят идеально быстро. На практике время переключения контекста зависит от архитектуры процессора, его тактовой частоты, типа, разрядности и тактовой частоты памяти, а также от качества используемой операционной системы.

Приведённые рассуждения могут показаться очевидными и банальными большинству разработчиков встраиваемых систем, однако в некоторых смежных технических областях до сих пор можно встретить мнение, что операционная система реального времени – это дорогостоящий и ненужный слой между процессором и приложением.

Во всех контроллерах FASTWEL I/O, начиная с 2008 года, поставляется система исполнения приложений CoDeSys 2.3 с вытесняющей многозадачностью. В оригинальной системе исполнения для 16-разрядных и небольших 32-разрядных встраиваемых микропроцессоров и микроконтроллеров, поставляемой фирмой 3S-Smart Software Solutions GmbH производителям ПЛК в составе комплекта адаптации системы исполнения CoDeSys 2.3, не предусматривался режим многозадачного исполнения приложений МЭК 61131-3. Многозадачный режим мог использоваться только в полномасштабной системе исполнения, рассчитанной на применение в контроллере полноценной операционной системы, подходящего 32-разрядного процессора и приличного объёма доступной оперативной памяти. Контроллеры модельного ряда CPM70x не подходили для развёртывания полномасштабной системы исполнения CoDeSys 2.3. Стоимость одной лицензии на такую систему (точнее, стоимость лицензионной марки, которая должна наклеиваться на каждый контроллер с CoDeSys) составляла значительную долю от ожидаемой



Условные обозначения: ***A*** – алгоритм; ***T*** – период цикла; t_1, t_2 – моменты времени; τ – время выполнения.

Рис. 7. Циклограмма исполнения алгоритмов разной срочности и длительности на контроллере с вытесняющей многозадачностью

полной стоимости процессорного модуля контроллера.

Кроме того, в оригинальной многозадачной системе исполнения CoDeSys 2.3 имеется одна особенность, о которой приходится помнить разработчикам приложений для контроллеров, чьи производители декларируют поддержку многозадачности: если в проекте имеется две задачи и более, программные единицы которых ссылаются на один и тот же адрес в области входных данных, то программные единицы, выполняющиеся в контексте менее приоритетных задач, вытесняемых более приоритетной задачей, могут функционировать непредсказуемо. Об этом стоит рассказать чуть подробнее, поскольку данная особенность не очень очевидна.

В CoDeSys 2.3 сегментом (или областью) входных данных является область памяти, находящаяся в распоряжении системы исполнения контроллера, в которой компилятор размещает значения всех переменных, объявленных со спецификатором доступа AT %I, то есть отображённых на входные каналы периферийных модулей и коммуникационных объектов, получаемых контроллером по сети.

Пусть в проекте CoDeSys 2.3 имеется программа CALC_PRG, выполняющаяся в контексте циклической задачи CalcTask с периодом 10 мс и вычисляющая значение на канале, доступном в сегменте входных данных приложения по адресу %IB2535.

Программа LOG_PRG выполняется под управлением второй циклической задачи LogTask с периодом 5 мс и в каждом цикле записывает текущее значение на том же входном канале в кольцевой буфер размером 16 слов.

Поскольку программа LOG_PRG выполняется с большей частотой, чем CALC_PRG, и требует меньшего времени выполнения за счёт отсутствия условных переходов и инструкций с плавающей точкой, задаче LogTask установлен более высокий приоритет, то есть LogTask может вытеснять CalcTask. Исходные тексты программ и конфигурация задач проекта CoDeSys 2.3 показаны на рис. 8.

Пусть на очередном цикле CalcTask программа CALC_PRG прочитала значение wChannelData и, убедившись, что оно отлично от нуля, перешла к началу вычисления выражения, не допускающего нулевого значения в знаменателе. Предположим, что в этот момент возникло некоторое событие, например прерывание системного таймера, повлекшее за собой вызов планировщика операционной системы контроллера, который выяснил, что пришло время «отдать» процессор задаче LogTask. Непосредственно перед вызовом LogTask происходит обновление сегмента входных данных из окружения, допустим, из канала модуля аналогового ввода или Holding-регистра MODBUS, причём там вновь оказалось нулевое значение, в результате чего слово по адресу %IB2535 вновь стало равным 0. Далее происходит вызов программы LOG_PRG, которая помещает нулевое значение в кольцевой буфер, завершает свой очередной цикл, после чего управление передаётся ранее вытесненной задаче CalcTask. Но задача CalcTask была прервана внутри CALC_PRG непосредственно перед началом вычисления выражения, не допускающего нулевого значения в знаменателе. Теперь при вычислении выражения программа CALC_PRG загружает нулевое значение из сегмента входных данных и пробует разделить 23,5 на 0,0, что явно не предусматривалось изначально.

В принципе, для переменных, отображаемых на адреса в сегменте вход-

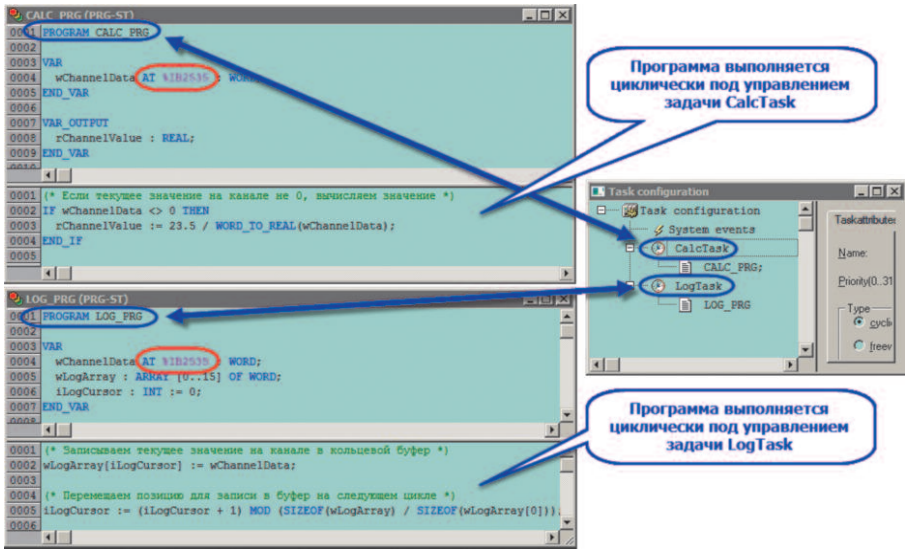


Рис. 8. Исходные тексты программ и конфигурация задач проекта CoDeSys 2.3 (две задачи ссылаются на один адрес в сегменте входных данных приложения)

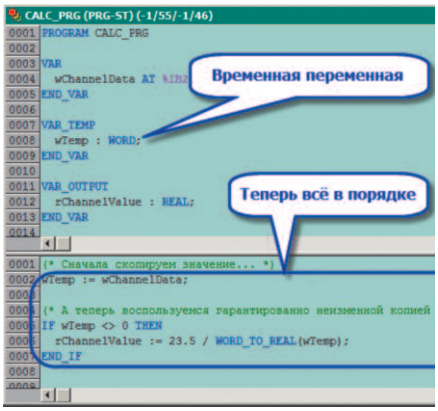


Рис. 9. Исходный текст программы с обеспечением неизменности значения входной переменной на очередном цикле

ных данных, размер которых не превышает естественной разрядности процессора (WORD для 16-разрядного и DWORD для 32-разрядного процессора), данная проблема может быть решена путём копирования данных в промежуточные переменные перед началом вычислений, как показано на рис. 9. Но если размер такой переменной превышает естественную разрядность процессора, что бывает при использовании переменных типа LREAL, структур и массивов, то нарушение целостности данных может произойти во время копирования, поскольку задача может быть вытеснена более приоритетной задачей в любой момент времени.

В контроллерах FASTWEL I/O указанная проблема решена на системном уровне двумя способами:

1. В контроллерах CPM70x, где может использоваться до трёх циклических задач, ввод данных из окружения в сегмент, на который ссылаются переменные при помощи директивы %I, производится отдельной системной высокоприоритетной задачей, которая в документации называется сервисной, и только тогда, когда все циклические задачи закончили свой очередной цикл.
2. В контроллерах CPM71x каждая из 16 циклических задач имеет собственный сегмент входных данных, то есть разные задачи никогда не могут нарушить целостность переменных, ссы-

лающихся на одни и те же адреса сегмента входных данных.

При наличии более одной циклической задачи в приложении, загруженном в контроллер CPM70x, сервисная задача функционирует с периодом, наименьшим среди всех циклических задач, и на очередном цикле атомарно устанавливает признак готовности обновить сегмент входных данных, после чего ожидает, когда все циклические задачи завершат свои очередные циклы. Каждая из циклических задач, завершив очередной цикл, обнаруживает признак готовности сервисной задачи к обновлению сегмента входных данных и приостанавливается, пока данные не будут обновлены.

Сервисная задача получает сигнал, когда все циклические задачи завершают свои очередные циклы, обновляет сегмент входных данных, что при максимальном количестве модулей ввода-вывода и Holding-регистров (принимаемых по сети коммуникационных протоколов CANopen или PROFIBUS) в конфигурации приложения занимает в среднем около 30 мкс, но не более 120 мкс, после чего сервисная задача сбрасывает ранее установленный признак своей готовности к обновлению сегмента и сигнализирует циклическим задачам, что они могут начинать свои очередные циклы. Если сразу нескольким циклическим задачам пришло время начинать очередные циклы, то процессор «получает» наиболее приоритетная из них, а если несколько задач имеют одинаковый приоритет, то первой начнёт выполняться та из задач, которая ожидала процессор дольше всех.

Из приведённого описания следует, что для сбалансированного функционирования всех задач приложения в пределах заданных для них периодов на контроллерах CPM70x минимальный период задачи с наивысшим приоритетом должен быть более максимального времени выполнения наименее приоритетной задачи, в противном случае низкоприоритетные задачи будут задерживать более приоритетные.

Принципиально иной механизм обеспечения целостности сегмента входных данных приложения используется в контроллерах CPM71x. Сразу после загрузки приложения в контроллер исполняемый код вместо реальных адресов переменных, которыми он оперирует, содержит ссылки на записи в таблице размещения переменных (relocation table), поскольку компилятор среды разработки CoDeSys 2.3 не знает, по каким физическим адресам в памяти контроллера будут расположены сегменты данных приложения. Каждая запись в таблице содержит номер сегмента и смещение в сегменте.

Система исполнения при запуске контроллера с загруженным в него приложением первым делом создаёт сегменты данных приложения: сегмент входных данных, сегмент выходных данных и сегмент глобальных и внутренних переменных, запоминает адреса созданных сегментов, а также адрес сегмента энергонезависимых переменных, который располагается в статической памяти с питанием от встроенной батареи.

Затем система исполнения путём анализа исполняемого кода строит списки программных единиц (программ, экземпляров функциональных блоков и функций), вызываемых из каждой циклической задачи.

Далее для каждой циклической задачи создаётся собственный сегмент входных данных, после чего выполняется замена

упомянутых ссылок на записи в таблице размещения переменных на физические адреса переменных, которые теперь стали известны. При обнаружении в коде приложения ссылки на сегмент входных данных выясняется, к какой программной единице относится текущий обрабатываемый адрес исполняемого кода. По номеру программной единицы и ранее построенным для каждой задачи спискам вызываемых из них программных единиц выясняется, из какой задачи вызывается программная единица с данным номером, после чего в сегмент кода записывается адрес переменной в собственном сегменте входных данных задачи, который вычисляется путём сложения адреса начала персонального сегмента входных данных задачи и смещения, полученного из записи в таблице размещения переменных.

Указанный подход позволяет полностью изолировать входные данные каждой задачи, получаемые из окружения, и требует, чтобы любая программная единица (программа, экземпляр функционального блока и функция), ссылающаяся на адреса в сегменте входных данных приложения, вызывалась только из одной задачи. Данное ограничение представляется не таким существенным, как необходимость использовать только одну задачу для ввода данных модулей ввода-вывода и сетевых коммуникационных объектов, что характерно для абсолютного большинства контроллеров разных производителей, декларирующих поддержку вытесняющей многозадачности в системе исполнения приложений CoDeSys версий 2.3 и 3.

На контроллерах CPM70x применить данный подход не удаётся, поскольку компилятор CoDeSys 2.3 для процессора 80186 размещает все переменные приложения в одной области памяти размером 64 кбайт, где располагаются сегменты входных и выходных данных, а также сегмент внутренних переменных, а все ссылки на переменные в коде приложения представлены 16-разрядными смещениями относительно начального адреса этой области. Таким образом, перед вызовом кода, сгенерированного компилятором CoDeSys 2.3, для правильного разрешения ссылок на переменные системе исполнения контроллера достаточно один раз загрузить значение сегментной составляющей начального адреса упомянутой области размещения переменных приложения в регистр DS процессора. Это значительно сокращает размер генерируемого исполняемого кода, поскольку при чтении и записи переменных не нужно формировать их полный адрес, а достаточно использовать инструкции, оперирующие смещениями переменных. Кроме того, сама таблица размещения переменных, формируемая компилятором CoDeSys 2.3, располагается в сегменте кода и имеет размер в несколько килобайт, что уменьшает размер части сегмента кода, доступной для размещения инструкций пользовательского приложения.

Таким образом, система исполнения приложений CoDeSys 2.3 в контроллерах FASTWEL I/O отличается от используемой в контроллерах других производителей тем, что в ней на системном уровне обеспечивается целостность входных данных для многозадачных приложений, разрабатываемых пользователями.

Автор – сотрудник ЗАО «НПФ «ДОЛОМАНТ»
Телефон: (495) 234-0639
E-mail: alexander.lokotkov@dolomant.ru

НОВОСТИ НОВОСТИ НОВОСТИ НОВОСТИ НОВОСТИ НОВОСТИ

ACME – вершина портативных компьютерных систем

В апреле этого года подписано партнёрское соглашение между корпорацией ACME PORTABLE (ACME) и компанией ПРОСОФТ. Таким образом, компания ПРОСОФТ стала официальным представителем ACME в России и СНГ.

Компания ACME основана в 1994 году, специализируется на разработке и производстве защищённых рабочих станций, компьютерных платформ с несколькими дисплеями, консолей оператора и заказных систем. Головной офис компании находится в Тайбэе на Тайване, производственные подразделения расположены также в Лос-Анжелесе (ACME Portable Machines, Inc.) и в Карлсруэ (ACME Portable Computers GmbH).

Акме (древнегреческое ακμή – высшая точка, вершина) – этим словом древние греки обозначали высшую стадию развития, и компания, выбравшая его своим названием, приоритетами считает революционный подход к компьютерной индустрии, великолепное качество продукции и отличный сервис. Производство мирового уровня, инжиниринг с прицелом на экономическую эффективность позволяют компании выпускать конкурентные и высокопроизводительные продукты, соответствующие требованиям MILSPEC, IEC, NEMA и др., поддерживать стандарты качества ISO-9001.

ACME, наверное, единственный производитель, предлагающий столь широкие возможности по адаптации своих готовых платформ, удовлетворяющих разнообразные требования пользователей по мощности, производительности, объёмам хранения данных и дополнительной периферии. Кроме этого, большой опыт и высокий технологический уровень компании позволили ACME стать партнёром многих всемирно известных фирм и поставщиком уникальных заказных платформ для специальных применений.

Портативные рабочие станции и операторские KVM-консоли ACME широко используются для построения систем измерений и тестирования, анализа сетей LAN/WAN и телекоммуникационных сигналов, теле- и радиовещания. В настоящее время они также весьма востребованы в качестве платформ для оборонных и нестандартных заказных устройств, выполненных по ТЗ пользователей.

Наличие в номенклатуре ПРОСОФТ продукции такого производителя, как ACME, позволит полнее удовлетворять запросы клиентов на поставку требуемого оборудования, а также предлагать собранные на заказ системы, оснащённые необходимыми периферийными платами, программным обеспечением и готовые к работе в клиентских проектах.