

Fastwel I/O изнутри

Александр Локотков

В статье рассматриваются внутреннее устройство и принципы функционирования основных составных частей аппаратно-программного комплекса Fastwel I/O, предназначенного для создания автоматизированных систем сбора данных и управления. Представлены подходы к проектированию и детально описаны межмодульная внутренняя шина FBUS, адаптированная среда исполнения прикладных программ CoDeSys, сервисы сетевых протоколов и особенности взаимодействия составных частей комплекса друг с другом.

Часть 3

ОСНОВНЫЕ ПОДСИСТЕМЫ КОНТРОЛЛЕРА FASTWEL I/O. АДАПТИРОВАННАЯ СРЕДА ИСПОЛНЕНИЯ ПРИКЛАДНЫХ ПРОГРАММ CODESYS

В данном разделе будет описан сервис исполнения прикладных программ контроллеров Fastwel I/O, реализованный на базе среды исполнения CoDeSys фирмы 3S-Smart Software Solutions.

Работа с Fastwel I/O подразумевает создание пользователем проекта прикладного программного обеспечения для каждого контроллера. Процесс разработки проекта в CoDeSys состоит из следующих операций:

- 1) создание проекта для платформы Fastwel I/O с выбором требуемого типа контроллера в редакторе конфигурации CoDeSys;
- 2) создание конфигурации модулей ввода-вывода контроллера;
- 3) создание конфигурации внешней сети, включая установку адреса узла, параметров обмена и создание списка описаний коммуникационных объектов сети;
- 4) разработка прикладной программы;
- 5) отладка прикладной программы в режиме эмуляции;
- 6) трансляция прикладной программы;
- 7) загрузка прикладной программы в контроллер;
- 8) мониторинг переменных и отладка прикладной программы в контроллере.

Программная модель контроллера

Пользователю ничего не известно о внутреннем устройстве контроллера, поскольку тот представлен для него некоторой программной моделью, определяющей порядок выполнения прикладной программы, систему типов данных и набор операций над данными, которые могут использоваться в программе для выполнения каких-либо полезных действий. Кроме того, программная модель определяет способ взаимодействия с устройствами ввода-вывода и внешней сетью, которые, в свою очередь, представляются моделью окружения программы.

Прикладная программа CoDeSys, разрабатываемая пользователем, представляется программой в терминах МЭК 61131-3 с именем PLC_PRG, из которой, в свою очередь, могут вызываться другие программы, функции и экземпляры функциональных блоков МЭК 61131-3, реализованных

на языках ST, CFC, SFC, LD, FBD, IL. Программа PLC_PRG, транслированная в исполняемый код процессора 80186 и загруженная в контроллер, вызывается средой исполнения CoDeSys с периодом, установленным в конфигурации контроллера. К сожалению, у фирмы 3S-Smart Software Solutions не имеется многозадачного варианта среды исполнения CoDeSys для процессора с архитектурой 80186, поэтому в Fastwel I/O пользователь не имеет возможности создавать несколько задач МЭК 61131-3 в проекте, разрабатываемом в среде CoDeSys.

Почему в предыдущем абзаце были упомянуты некие экземпляры функциональных блоков? Потому что функциональный блок, строго говоря, есть не что иное, как абстрактный тип данных, снабжённый функцией, которая реализует некоторый пользовательский алгоритм. В прикладных программах может быть создано и использовано множество экземпляров одного и того же функционального блока.

Перечисленные элементы программной модели, относящиеся к исполняемой части программы, согласно МЭК 61131-3 называются единицами организации программы (Program Organization Units — POU). В адаптированном варианте среды исполнения CoDeSys для Fastwel I/O максимальное количество единиц организации программы ограничено 1024.

Единица организации программы при вызове получает входные данные, обрабатывает их, в том числе с учётом своего внутреннего состояния, и передаёт результаты в виде выходных данных другим единицам организации программы или во внешнее окружение.

Входные данные программы, функционального блока и функции представляются так называемыми входными переменными, которые объявляются в секции VAR_INPUT области декларации переменных каждой POU в редакторе программ.

Выходные данные программы, функционального блока и функции представляются так называемыми выходными переменными, которые объявляются в секции VAR_OUTPUT области декларации каждой POU.

Внутреннее состояние единицы организации программы представляется так называемыми внутренними переменными, которые объявляются в секции VAR области декларации переменных редактора программ. При этом программа и эк-

земляр функционального блока, в отличие от функции, сохраняют свое состояние между вызовами. Иными словами, внутренние переменные программы и экземпляра функционального блока сохраняют значения между вызовами.

В адаптированной среде CoDeSys для Fastwel I/O поддерживаются следующие примитивные типы МЭК 61131-3: BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL и TIME. К непримитивным типам относятся массив (ARRAY) и структура (STRUCT). Специальными типами являются указатель (POINTER TO), перечисление, поддиапазон, ссылка и строка (STRING). Более подробная информация приведена в документации по CoDeSys.

Размер области памяти, отведённой под размещение внутренних переменных POU и глобальных переменных, доступных всем POU (и использование которых, по мнению автора, крайне нежелательно!) в адаптированном варианте среды исполнения CoDeSys для Fastwel I/O, составляет 24576 байт. В этой области компилятор среды разработки CoDeSys размещает все переменные состояния всех единиц организации программы и, к сожалению, все константы и временные переменные (переменные, объявляемые в секциях VAR_TEMP). Почему мы остановились на таком размере памяти внутренних переменных? Компилятор CoDeSys для 80186 обеспечивает возможность размещения входных, выходных, глобальных и внутренних переменных в единственном сегменте процессора, размер которого не может превышать 64 кбайт. В этом же сегменте компилятор располагает некоторую служебную информацию, поэтому нам пришлось искать некоторое более или менее сбалансированное соотношение размеров областей с учётом возможностей контроллера по вводу-выводу, а также по сложности и размеру разрабатываемых программ.

Сохраняемые (VAR RETAIN) и не изменяемые после «горячей» перезагрузки программы (VAR PERSISTENT) переменные нашей адаптации CoDeSys в настоящее время не поддерживаются. Мы также исключили поддержку флагов, которые могут задаваться для некоторых переменных программы (poinit, nowatch и т.д.).

Исключение поддержки флагов обусловлено необходимостью экономить память, так как компилятор CoDeSys для 80186 размещает и адресует эту информацию в том же сегменте, где находятся входные, выходные и внутренние переменные.

Отсутствие поддержки сохраняемых (VAR RETAIN) переменных в текущей версии объясняется следующими соображениями. VAR RETAIN представляют стандартный по МЭК 61131-3 механизм автоматического сохранения некоторой части значений переменных программ в энергонезависимой памяти. Семантика VAR RETAIN состоит в том, что значения переменных, помещённых в данную секцию, должны сохраняться в энергонезависимой памяти контроллера и загружаться из неё даже после внезапного отключения питания.

Возможны два основных варианта технической реализации сохраняемых переменных:

- 1) сохранение переменных области VAR RETAIN только в момент пропадания питания контроллера и восстановление при последующем включении;
- 2) сохранение и восстановление области VAR RETAIN переменных между циклами программы, содержащей данную область.

При использовании обоих способов в состав контроллера должна входить быстрая энергонезависимая память, размера которой достаточно для размещения области сохраняемых переменных.

Насколько «быстрой» должна быть данная память? Например, типовая программа для контроллера Fastwel I/O, опрашивающая дюжину модулей ввода-вывода, спокойно укладывается в цикл 1 мс. Время записи любого количества данных размером до 8 кбайт во встроенную NAND флэш-память составляет 15-20 мс. Поэтому необходима либо статическая память (SRAM) с питанием от батарейки, либо ферроэлектрическая память с произвольным доступом (FRAM).

Статическая память с питанием от батарейки означает неизбежный выход из строя батарейки в самый неподходящий момент. При этом пользователю придётся либо самостоятельно заменять батарейку, либо везти контроллер в ремонт.

FRAM прекрасно подходит по скорости доступа (250 нс), однако имеет ограниченное количество циклов записи (от нескольких миллионов до 10 миллиардов). Что же такое 10 миллиардов циклов достоверной записи, с точки зрения программы контроллера? Если считать, что запись производится между двумя соседними циклами программы, срок службы контроллера в часах составит около $2700 \times T_{mc}$, где T_{mc} — период вызова программы в миллисекундах. То есть при периоде, равном 10 мс, память будет изношена примерно через 3 года.

Таким образом, непосредственное использование второго варианта реализации механизма VAR RETAIN приемлемо далеко не для всех возможных применений контроллера. Иными словами — неприемлемо. Возможны компромиссные решения.

1. Указать в документации, что сохраняемые переменные могут изменяться только очень редко (уставки и коэффициенты регуляторов и т.п.), и перезаписывать данные только по изменению. Но вообще говоря, формально ничем не гарантируется, что они не начнут изменяться в каждом цикле. В ряде случаев уставки могут изменяться в каждом цикле в соответствии с алгоритмом.
2. Перезаписывать по изменению и «не чаще чем», оговаривая величину «не чаще чем» в документации. Такой вариант непригоден для ситуаций, когда в программе имеются адаптивные регуляторы, которые на определённых фазах управления процессом пересчитывают свои коэффициенты и уставки в каждом цикле контроллера. Кроме того, пользователь будет вынужден обеспечивать сохранность переменных из области VAR RETAIN специальными средствами поддержки питания (источник бесперебойного питания и т.п.).

Теперь о сохранении области VAR RETAIN только в момент пропадания питания с помощью обработчика прерывания, формируемого аппаратным монитором питания. Встроенный преобразователь питания контроллеров Fastwel I/O рассчитан на максимальный ток потребления 1,2 А. Время, необходимое для копирования 8 кбайт данных из оперативной памяти в FRAM на процессоре R1610C, составит около 400 мкс. Диапазон значений напряжения, в пределах которого сохраняется работоспособность основных подсистем вычислительного ядра контроллера с момента срабатывания монитора питания при пропадании внешнего напряжения, составляет около 5% от номинала. Тогда ёмкость конденсатора, устанавливаемого в выходной цепи вторичного преобразователя питания контроллера, должна составлять около 1800 мкФ. Два танталовых конденсатора, с некоторым запасом образующих требуемую ёмкость, займут площадь около 25×20 мм, что практически неприемлемо при использовании конструктива WAGO. В приведённых рассуждениях не учитывается тот факт, что разные подсистемы вычислительного ядра имеют разные номинальные значения напряжения пи-

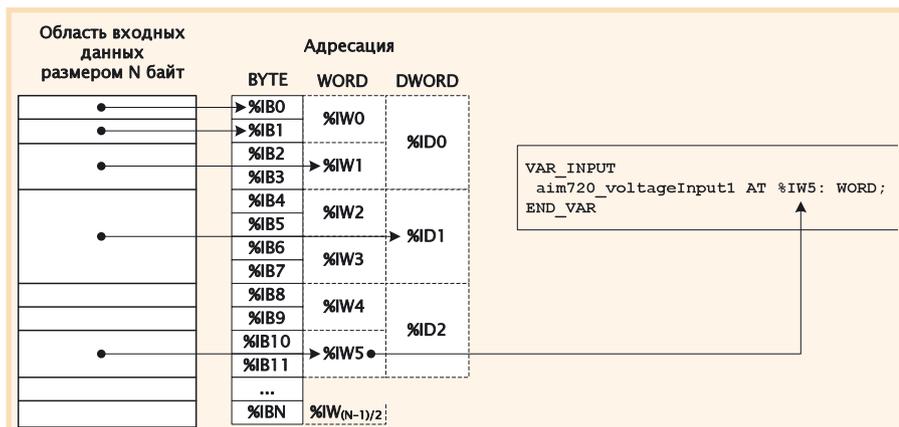


Рис. 15. Адресация области входных данных и связывание с входной переменной

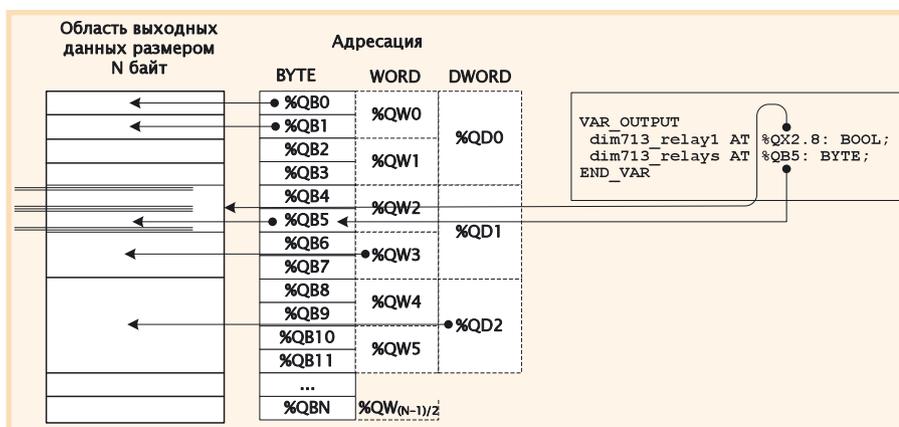


Рис. 16. Адресация области выходных данных и связывание с выходной переменной

тания, что существенно усложняет реализацию монитора питания. Наконец, данная система должна срабатывать только при выключении питания (скорость нарастания напряжения при включении питания должна быть максимальной) и сохранять работоспособность в диапазоне от -40 до +85°C, что делает систему питания еще сложнее.

Таким образом, было принято решение в первых версиях контроллеров отказаться от реализации механизма VAR RETAIN на системном уровне в надежде, что конкретные потребности пользователей позволят нам выбрать наиболее подходящий для большинства применений вариант реализации.

Причины, по которым пока не поддерживаются неизменяемые (VAR PERSISTENT) переменные, будут озвучены далее при описании механизмов взаимодействия среды исполнения контроллера со средой разработки CoDeSys.

Модель окружения

Модель окружения программы представляется так называемым образом процесса, который состоит из двух областей памяти с непересекающимися адресами. Первая область образа процесса, называемая областью входных данных, предназначена для хранения и обновления значений входных данных прикладной программы в процессе приёма информации от устройств ввода-вывода и сетевых интерфейсов. Вторая область называется областью выходных данных и предназначена для хранения и обновления значений выходных данных программы в процессе выдачи информации устройствам ввода-вывода и в сетевые интерфейсы. Размеры областей входных и выходных данных среды исполнения CoDeSys для Fastwel I/O составляют по 12800 байт каждая.

Для ввода данных из окружения в программе могут быть определены так называемые непосредственно представля-

емые входные переменные, ссылающиеся на адреса в области входных данных. Например:

```

VAR_INPUT
aim720_voltageInput1 AT %IW5: WORD;
END_VAR
    
```

В данном примере декларирована входная переменная с именем aim720_voltageInput1 типа WORD, которая ссылается на 6-е слово в области входных данных, как показано на рис. 15.

Для вывода данных в окружение в программе могут быть определены так называемые непосредственно представляемые выходные переменные, ссылающиеся на адреса в области входных данных. Например:

```

VAR_OUTPUT
dim713_relay1 AT %QX2.8: BOOL;
dim713_relays AT %QB5: BYTE;
END_VAR
    
```

В данном примере декларированы выходные переменные dim713_relay1 типа BOOL и dim713_relays типа BYTE, которые ссылаются на 8-й бит 3-го слова и 6-й байт в области выходных данных, как показано на рис. 16. Следует взять на заметку, что при наличии нескольких выходных переменных, ссылающихся на

один и тот же адрес в области выходных данных в большой программе, довольно легко не заметить ошибочное «лишнее» присвоение по выходному адресу неправильного, с точки зрения прикладного алгоритма, значения. К счастью, компилятор CoDeSys умеет обнаруживать множественные ссылки на один и тот же адрес в области выходных данных программы и предупреждать об этом разработчика.

Адреса каналов ввода-вывода и полей коммуникационных объектов внешней сети создаются автоматически средой разработки при добавлении описаний модулей и коммуникационных объектов в конфигурацию контроллера в секции PLC Configuration, как показано на рис. 17.

Следует отметить, что вовсе не обязательно явно объявлять в программах входные и выходные непосредственно представляемые переменные, ссылающиеся на адреса каналов модулей ввода-вывода и поля данных коммуникационных объектов внешней сети, поскольку в CoDeSys (и в МЭК 61131-3) имеется возможность использования адресов окружения в качестве правой части и операндов выражений:

```

someVariable := DWORD_TO_REAL(%IB27) * Koeff;
%QX2.8 := NOT toggleBit;
    
```

Однако это не очень хорошая идея, поскольку если по какой-то причине изменится структура образа процесса программы, например при изменении типа какого-нибудь модуля ввода-вывода в конфигурации контроллера, придётся вносить в программу очень много изменений. А значит, в программе неизбежно появятся ошибки, о которых компилятор известит программиста далеко не всегда.

Кроме того, мы пока отключили поддержку задания имён непосредственно представляемых переменных в дереве PLC Configuration. Это обусловлено следующими соображения-

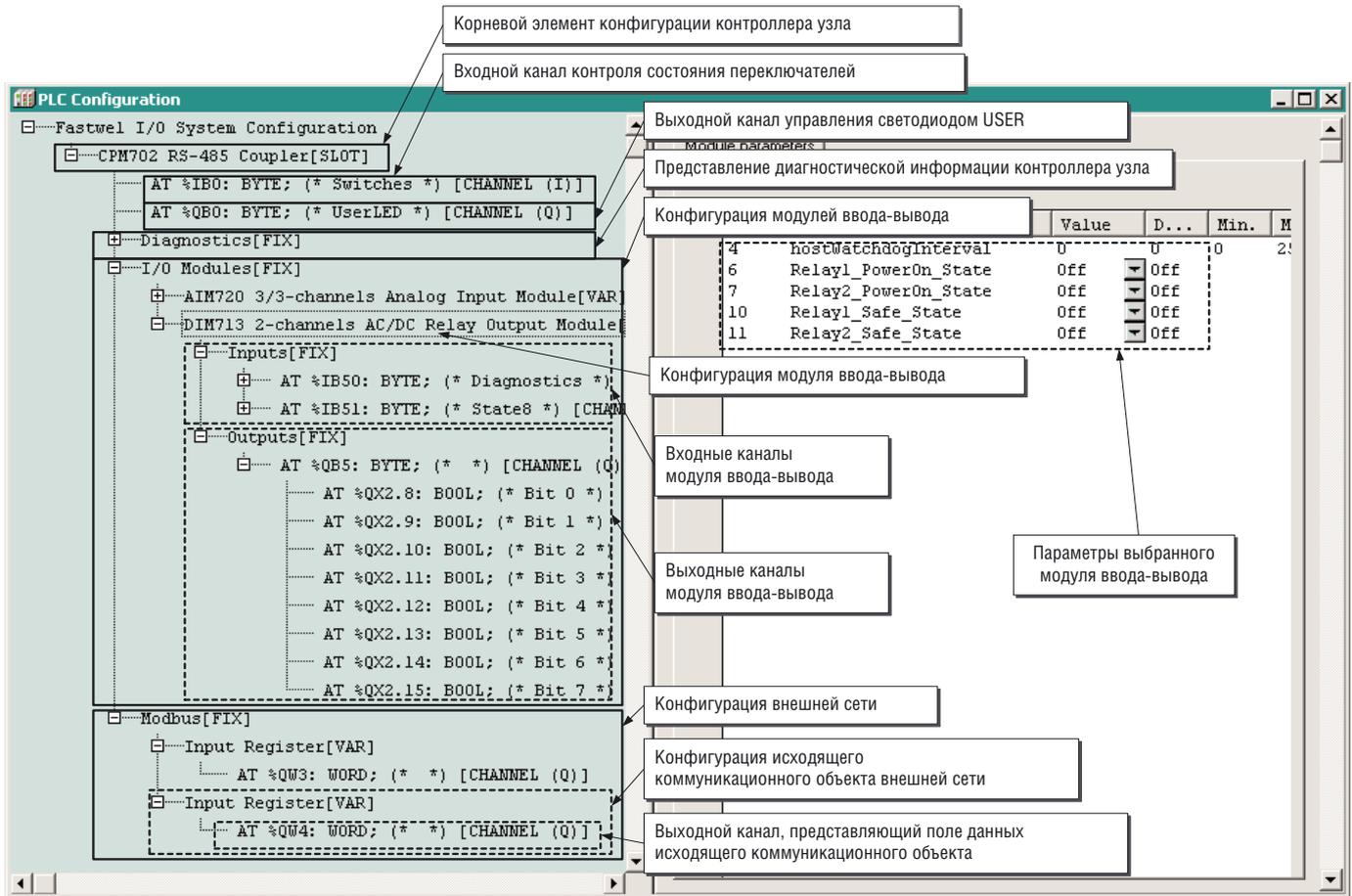


Рис. 17. Элементы конфигурации контроллера

ми. По нашему мнению, жёсткая связь между каналами ввода-вывода и переменными прикладных программ приемлема только для очень простых случаев и для контроллеров с небольшим фиксированным количеством каналов ввода-вывода. Fastwel I/O является модульной системой, в состав которой может входить до 64 модулей с самой разнообразной номенклатурой каналов ввода-вывода, поэтому о небольшом и тем более фиксированном наборе каналов говорить не приходится. Итак, входные и выходные переменные программы не должны «прибиваться гвоздями» к каналам конкретных устройств ввода-вывода и коммуникационных объектов внешней сети по следующим причинам:

1. В процессе проектирования некоторой системы уже после написания программы может потребоваться изменить тип модулей ввода-вывода. Например, модуль аналогового ввода одного типа после опытной эксплуатации заменяется на другой. В конфигурации контроллера модулей заменяемого типа может быть до 64, и пользователю придется вручную «прибивать» идентификаторы переменных для всех модулей повторно, что весьма трудоёмко и ведёт к ошибкам.
2. Утрачивается возможность структурной обработки данных от однотипных модулей, как продемонстрировано далее. Мало того, что при жёсткой связи переменных с каналами нужно при помощи мыши «объявлять» каждую входную-выходную переменную в PLC Configuration, так потом ещё в коде придётся вручную задавать каждую переменную блоку или функции обработки в качестве фактических параметров, что увеличивает количество кода и вероятность ошибок. В небольших проектах с небольшим количеством устройств ввода-вывода, коммуникационных объектов и с небольшими программами указанная проблема не будет столь заметной, но для больших проектов структурная об-

работка входных и выходных данных даёт выигрыш по времени на пару порядков, а то и больше. Плюс во много раз сокращается количество кода обработки.

3. При перераспределении алгоритмической функциональности между разными контроллерами (например, был один контроллер, и в связи с введением новых очередей системы управления решили добавить второй и поделить между ними возросшую функциональность) может потребоваться получать значения входных переменных не от модулей ввода-вывода, а по сети. При использовании структурной обработки будет достаточно изменить только начальный адрес блока объявления массива входных переменных типа STRUCT, скажем, «наведя» его на поля данных первого из нескольких смежных однотипных коммуникационных объектов внешней сети, посредством которых предполагается получать данные от другого узла сети. Рассмотрим пример, который иллюстрирует технику структурной обработки входных и выходных данных.

Пусть, к примеру, в конфигурацию контроллера добавлены 6 модулей аналогового ввода типа AIM726 и 9 модулей аналогового ввода типа AIM728, причём однотипные модули располагаются в конфигурации друг за другом. Предположим, что предусмотрительный пользователь определил в программе непримитивные типы данных, описывающие входные и выходные каналы модулей следующим образом:

```
(* Входы AIM726 *)
TYPE AIM726_inputs :
STRUCT
    diagnostics: BYTE;
    vin0: DWORD;
    vin1: DWORD;
END_STRUCT
END_TYPE
(* Значения напряжения на входах AIM726 *)
```

```
TYPE aim726_outputs :
  STRUCT
    vout0: REAL;
    vout1: REAL;
  END_STRUCT
END_TYPE
```

(* Входы AIM728 *)

```
TYPE AIM728_inputs :
  STRUCT
    diagnostics: BYTE;
    vin0: DINT;
    vin1: DINT;
    vin2: DINT;
    vin3: DINT;
  END_STRUCT
END_TYPE
```

(* Значения напряжения на входах AIM728 *)

```
TYPE aim728_outputs :
  STRUCT
    vout0: REAL;
    vout1: REAL;
    vout2: REAL;
    vout3: REAL;
  END_STRUCT
END_TYPE
```

Кроме того, пусть в проект добавлены два типа функциональных блока, предназначенных для преобразования результатов аналого-цифрового преобразования сигналов на каналах модулей:

(* Блок преобразования показаний на каналах AIM726 *)

```
FUNCTION_BLOCK AIM726_STIN
  VAR_INPUT
```

```
    inputs: AIM726_inputs;
  END_VAR
  VAR_OUTPUT
    (* Признак достоверности преобразования *)
    valid: BOOL;
    outputs: aim726_outputs;
  END_VAR
  (* Если значение на канале диагностики не равно 16#FF, то все в порядке *)
  valid := inputs.diagnostics <> 16#FF;
  IF valid THEN
    outputs.vout0 := DWORD_TO_REAL(inputs.vin0) * 4.7683729E-006;
    outputs.vout1 := DWORD_TO_REAL(inputs.vin1) * 4.7683729E-006;
  END_IF
END_FUNCTION_BLOCK
(* Блок преобразования показаний на каналах AIM728 *)
FUNCTION_BLOCK AIM728_STIN
  VAR_INPUT
    inputs: AIM728_inputs;
  END_VAR
  VAR_OUTPUT
    (* Признак достоверности преобразования *)
    valid: BOOL;
    outputs: aim728_outputs;
  END_VAR
  (* Если значение на канале диагностики не равно 16#FF, то все в порядке *)
  valid := inputs.diagnostics <> 16#FF;
  IF valid THEN
    outputs.vout0 := DINT_TO_REAL(inputs.vin0) * 2.3841861E-006;
    outputs.vout1 := DINT_TO_REAL(inputs.vin1) * 2.3841861E-006;
    outputs.vout2 := DINT_TO_REAL(inputs.vin2) * 2.3841861E-006;
    outputs.vout3 := DINT_TO_REAL(inputs.vin3) * 2.3841861E-006;
  END_IF
END_FUNCTION_BLOCK
```

Пара замечаний:

1. Поля типов AIM72x_inputs и AIM72x_outputs, соответствующие однотипным каналам (например vout0...3), могут быть описаны массивами.
2. Коэффициенты преобразования значений АЦП могут быть объявлены в виде констант в секции VAR CONSTANT. К сожалению, это решение, являющееся хорошим тоном в языках программирования общего применения, не очень подходит для данного случая. Это связано с тем, что компилятор CoDeSys размещает одни и те же константы в памяти каждого экземпляра РОУ. Более того, если требуется задать инициализирующее значение некоторой переменной в секции VAR, оно тоже займет место в памяти каждого экземпляра РОУ. А у нас размер области памяти, отводимой под переменные, составляет всего 24 кбайт.

Пусть, кроме того, требуется выводить в сеть Modbus значения напряжения на каналах модулей AIM726 и AIM729. Суммарное количество каналов составляет $2 \times 6 + 4 \times 9 = 48$, а значит, в конфигурации сети контроллера для передачи 48 значений типа REAL должно быть создано не менее 96 входных регистров со смежными идентификаторами (адресами). Пусть первый из 96 созданных регистров имеет адрес %QB7 в области выходных данных среды исполнения.

Программа, преобразующая показания 6 модулей аналогового ввода типа AIM726, 9 модулей аналогового ввода типа AIM728 и выводящая результаты в Modbus, может выглядеть следующим образом:

```
PROGRAM PLC_PRG
  VAR CONSTANT
    AIM726_ARRAY_SIZE :INT := 5;
    AIM728_ARRAY_SIZE :INT := 8;
    NETWORK_BUF_BOUND :INT := 47;
  END_VAR
  VAR_INPUT
    (* Адрес первого канала первого модуля AIM726 из шести - %IB37 *)
    aim726_inputs AT%IB37 : ARRAY [0..AIM726_ARRAY_SIZE] OF
      AIM726_inputs;
    (* Адрес первого канала первого модуля AIM728 из девяти - %IB91 *)
    aim728_inputs AT%IB91 : ARRAY [0..AIM728_ARRAY_SIZE] OF
      AIM728_inputs;
  END_VAR
  VAR_OUTPUT
    (* Адрес канала первого регистра из 96-ти - %QB7 *)
    networkBuffer AT%QB7 : ARRAY [0.. NETWORK_BUF_BOUND] OF REAL;
  END_VAR
  VAR
    aim726_conv : ARRAY [0..AIM726_ARRAY_SIZE] OF AIM726_STIN;
    aim728_conv : ARRAY [0..AIM728_ARRAY_SIZE] OF AIM728_STIN;
    i : INT;
    netBufferIndex : INT;
  END_VAR
  (* Исполняемый код начинается здесь *)
  netBufferIndex := 0;
  FOR i := 0 TO AIM726_ARRAY_SIZE DO
    aim726_conv[i](inputs:= aim726_inputs[i], valid=> , outputs=> );
    networkBuffer[netBufferIndex] := aim726_conv[i].outputs.vout0;
    netBufferIndex := netBufferIndex + 1;
    networkBuffer[netBufferIndex] := aim726_conv[i].outputs.vout1;
    netBufferIndex := netBufferIndex + 1;
  END_FOR;
  FOR i := 0 TO AIM728_ARRAY_SIZE DO
    aim728_conv[i](inputs:= aim728_inputs[i], valid=> , outputs=> );
    networkBuffer[netBufferIndex] := aim726_conv[i].outputs.vout0;
    netBufferIndex := netBufferIndex + 1;
    networkBuffer[netBufferIndex] := aim726_conv[i].outputs.vout1;
```

```
netBufferIndex := netBufferIndex + 1;
networkBuffer[netBufferIndex] := aim726_conv[i].outputs.vout2;
netBufferIndex := netBufferIndex + 1;
networkBuffer[netBufferIndex] := aim726_conv[i].outputs.vout3;
netBufferIndex := netBufferIndex + 1;
END_FOR;
END_PROGRAM;
```

Как видно из приведённого исходного текста, в программе объявлены два массива входных непосредственно представляемых переменных типа AIM726_inputs и AIM728_inputs. Массив aim726_inputs, состоящий из шести элементов типа AIM726_inputs, размещается, начиная с адреса первого канала первого модуля AIM726 из шести имеющихся в конфигурации контроллера. Массив aim728_inputs, состоящий из девяти элементов типа AIM728_inputs, размещается, начиная с адреса первого канала первого модуля AIM728 из девяти имеющихся в конфигурации контроллера. Кроме того, для вывода в Modbus в программе объявлен массив из 48 переменных типа REAL, которые ссылаются на область выходных данных прикладной программы, начиная с адреса %QB7, то есть с того места, где располагается выходной канал первого из 96 смежных регистров.

Далее в программе объявлены шесть и девять массивов функциональных блоков типа AIM726_STIN и AIM728_STIN соответственно. Вызовы блоков преобразования выполняются в двух циклах. Теперь в случае добавления каких-либо модулей перед первыми шестью AIM726 достаточно будет скорректировать значения адресов, на которые ссылаются переменные-массивы aim726_inputs и aim728_inputs, заглянув в секцию PLC Configuration. Если же какие-нибудь модули вставляются между первыми шестью AIM726 и группой из девяти AIM728, нужно будет скорректировать значение адреса, на который ссылается переменная-массив aim728_inputs. Кроме того, имеется возможность считывать значения всех 48 аналоговых каналов за один запрос чтения группы регистров, передаваемый мастером Modbus контроллеру.

При использовании подобных приёмов следует учитывать, что они работают только тогда, когда однотипные объекты окружения (модули ввода-вывода или коммуникационные объекты) располагаются в конфигурации контроллера друг за другом. Кроме того, среда исполнения CoDeSys должна поддерживать чтение/запись по произвольному адресу без обязательного выравнивания на естественную границу для процессора, на базе которого сделан контроллер.

И еще одно замечание. В приведённом примере совершенно не обязательно объявлять и использовать в программе два массива экземпляров функциональных блоков AIM726_STIN и AIM728_STIN, вполне достаточно обойтись одним экземпляром каждого блока. Это связано с тем, что в данной программе используются только вычисляемые функции блоков. Применение отдельных экземпляров функциональных блоков требуется тогда, когда необходимо хранить состояние каждого объекта, представляемого каждым экземпляром блока, между вызовами программы. ●

Автор — сотрудник фирмы Fastwel
119313, Москва, а/я 242
Тел.: +7 (495) 234-0639
Факс: +7 (495) 232-1654
E-mail: info@fastwel.ru
Web: www.fastwel.ru