И. Петров

Язык ST для С программиста

Принято считать, что языки программирования ПЛК стандарта МЭК 61131-3 (далее для краткости МЭК) предназначены для применения неспециалистами в области информатики. Но возможности ПЛК растут, соответственно растут и требования, предъявляемые к прикладному программному обеспечению (ПО). Все чаше мы встречаем системы на ПЛК, включающие сложнейший математический аппарат. В ПЛК используют web-технологии, нечеткие регуляторы, многомерные интерполяторы систем управления движением (SoftMotion) и т.д. Такие технологии требуют специальных знаний. Несмотря на все прогнозы и рассуждения, на сегодняшний день более половины ПО для ПЛК создается профессиональными программистами, имеющими высшее образование. С каждым годом их доля возрастает.

Не смотря на это, подавляющее большинство изготовителей МЭК систем упорно продолжают создать простые инструменты «для чайников». Чаще всего это некий редактор FBD и интерпретирующая система исполнения, опирающаяся на вызов заранее откомпилированных модулей. Для расширения состава блоков их пишут на языке С внешними средствами. Если профессиональный программист сталкивается с такой средой, то часто результатом становится МЭК программа, состоящая из одного большого функционального блока, полностью написанного на С.

Стандарт МЭК включает высокоуровневый язык ST, по выразительной мощности не уступающий языкам С и Паскаль. К сожалению, в большинстве систем МЭК программирования он не реализован вовсе либо имеет массу ограничений. Как правило, это ограничение числа поддерживаемых типов данных, упрощенный интерпретирующий транслятор или промежуточный код, ограничения на доступ к аппаратным ресурсам, невозможность обрабатывать прерывания, выполнять динамическое распределение памяти, управлять процессами и др. На самом деле, это не ограничения языка ST. Это именно ограничения среды программирования.

Некоторым особняком в ряду комплексов МЭК стоит CoDeSys производства 3S-Smart Software Solution GmbH. Его уникальность состоит в том, что он изначально задумывался как инструмент для профессионалов. Исходя из собственных соображений, при адаптации CoDeSys для конкретного ПЛК, изготовитель может наложить ряд ограничений на возможности пользователя. Но изначально их нет.

Безусловно, в CoDeSys можно использовать внешние библиотеки, реализованные на С, но необходимость в этом возникает крайне редко. Практически все, что можно написать на С, в CoDeSys можно написать на ST, без изменения методологии проектирования. CoDeSys имеет встроенный компилятор и генераторы машинного кода для всех популярных семейств микропроцессоров. Внешний С компилятор не даст выигрыша по быстродействию, но значительно усложнит разработку.

Очень существенная тонкость стандарта МЭК состоит в том, что он включает только требование однозначности выполнения описанных в нем конструкций. Многие воспринимают это как возможность называть МЭК системой программирования среду, поддерживающую только часть стандарта. Однако справедлив и обратный подход: МЭК система может включать все, что описано в стандарте, плюс дополнительные возможности.

Практика показывает что, профессиональные программисты, работая с CoDeSys, чаше всего выбирают язык ST. Практически все они в совершенстве владеют языком С. Данная статья имеет целью быстро очертить С программистам границы возможностей языка ST в CoDeSys и подсказать типовые методы преодоления затруднений.

Типы данных.

В CoDeSys реализованы все стандартные типы данных МЭК 61131-3. Все типы данных доступны во всех 6 поддерживаемых языках. Специальных расширений для ST нет.

Целочисленные типы таковы: SINT (char), USINT (unsigned char), INT (short int), UINT (unsigned int), DINT (long), UDINT (unsigned long), LINT (64 бит целое), ULINT (64 бит целое без знака).

Действительные типы: REAL (float), LREAL (double).

Специальные типы BYTE, WORD, DWORD, LWORD представляют собой битовые строки длиной 8, 16, 32 и 64 бит соответственно. Битовых полей в ST нет. К битовым строкам можно непосредственно обращаться побитно. Например: a.3 := 1; (* Установить бит 3 переменной a *).

В С в подобных целях используются целые без знака и битовые логические операции. В CoDeSys к битовым стокам можно применять операции, доступные для целых без знака.

Логический тип BOOL. Может иметь значение TRUE или FALSE. Физически переменная типа BOOL может соответствовать одному биту. Обычно так и есть, если речь идет о дискретном входе либо выходе ПЛК или прямоадресуемой переменной (см. ниже). В иных случаях CoDeSys выделяет один байт. Это вызвано тем, что большинство микропроцессоров не умеют непосредственно адресоваться к отдельным битам памяти.

Строки STRING. Являются именно строкой, а не массивом. Поэтому (в версии V2.3) нет возможности обращаться к отдельным символам. Зато можно сравнивать и копировать строки стандартными операторами. Например так: strA := strB. В МЭК есть стандартный набор функций для работы со строками. Внутренний формат строк не стандартизован. CoDeSys использует нуль-терминированные строки.

Специальные типы в стандарте МЭК определены для длительности (TIME), времени суток (TOD), календарной даты (DATE) и временного штампа (DT). Работа со временем и интервалами встречается в ПЛК программах повсеместно. Достаточно подробно применение этих типов данных описано в [1].

Применение структур (STRUCT) не отличается от С. Описание структуры должно предшествовать объявлению переменной данного типа. Допускаются вложенные структуры и массивы.

Массивы (ARRAY) строятся из элементов любых типов, включая структуры и массивы. Из особенностей нужно отметить возможность задания повтора значений при инициализации. Например: bX ARRAY[0..20] OF BOOL := TRUE, 10(FALSE), 9(TRUE); Значение FALSE повторено здесь 10 раз и значение TRUE, соответственно 9 раз. Массивы (и структуры) можно копировать с помощью обычной операции присваивания: bY := bX;

Перечисления аналогичны перечислениям С. Пример: TYPE TEMPO: (Adagio := 1, Andante := 2); END_TYPE.

На базе целых можно построить типы, имеющие ограниченный диапазон значений. Например: TYPE DAC10: INT (0..16#3FF); END_TYPE.

Для любого типа можно создать псевдоним (typedef в С). Например: TYPE DEGR : UINT END_TYPE.

Константы.

В МЭК используются типизированные константы. Например: 2#10001000 (целое, байт в двоичном формате), INT#40, t#10h14m5s (время дня), D#2006-01-31 (дата). Формат констант описан в документации и не вызывает сложностей.

Символьные константы можно объявлять локально или глобально в специальной секции раздела объявлений: VAR CONSTANT.

Идентификаторы.

Идентификаторы имен переменных могут иметь любую длину. Ограничения числа значимых символов нет. Идентификаторы не чувствительны к регистру (XZ и Xz одно и тоже). Идентификаторы не должны начинаться с цифры (12x). Двойные подчеркивания запрещены (__X).

Инициализация переменных.

По умолчанию, все переменные инициализируются нулем. Иное значение переменной можно указать явно при ее объявлении. Например: str1: STRING := ‘Hello world’.

Переменные можно объявить как RETAIN или PERSISTENT. Это означает требование их размещения в энергонезависимой памяти контроллера (если она есть). Их инициализация производится только при первом запуске программы или по специальной команде.

Прямая адресация.

Помимо общей памяти данных в контроллере предусмотрены 3 специальные области памяти. Это область входов, выходов и прямо-адресуемая область. Эти области служат для связи системы исполнения контроллера и пользовательской программы. В простейших ПЛК распределение памяти в области входов/выходов фиксировано изготовителем. Мы можем непосредственно обращаться к нужному адресу из программы (например, %QB5 дает 5-й байт в области выходов) или объявить переменную, расположенную в соответствующем месте. Как правило, изготовители ПЛК снабжают CoDeSys конфигурационными файлами. Встроенный графический конфигуратор позволяет настроить и автоматически объявить переменные для всех

входов/выходов и диагностических переменных без программирования, включая распределенные модульные системы (См. рис. 1).

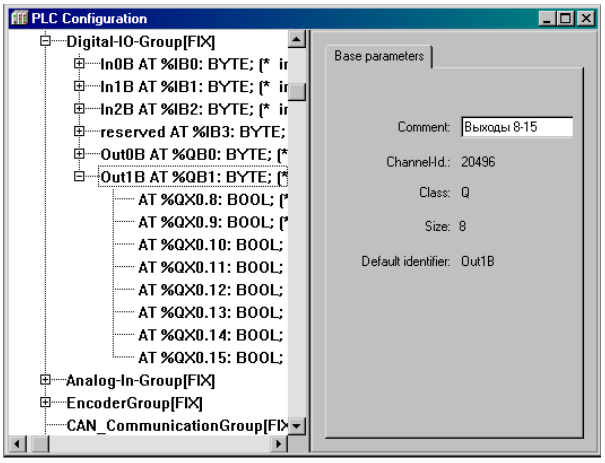

Рис. 1. Конфигурирование переменных входов/выходов в CoDeSys

Преобразование типов.

Неявное преобразование типов запрещено. Любое преобразование нужно делать явно с помощью специальных операторов. Запомнить их легко. Пишем исходный тип, далее _TO_ и нужный тип. Например: iX := REAL_TO_INT(rX); .

Операторы и функции.

Наиболее часто используемые операторы и функции ST приведены в таблице 1. В МЭК определен целый ряд общепринятых и специализированных (для ПЛК программ) функций и функциональных блоков. В их числе таймеры, триггеры, счетчики, регуляторы и многое другое.

Операция	**ST**
Присваивание	:=
Алгебраические операции	+, -, *, /
Замена знака	-
Остаток от деления	MOD
Абсолютное значение	ABS
Сравнения	< , > ,<=, >=
Равенство, неравенство	=, <>
И, ИЛИ, НЕ	AND, OR, NOT
Исключающее ИЛИ	XOR
Побитный сдвиг влево, вправо	SHL(in,n), SHR(in,n)
Циклический сдвиг влево, вправо	ROL(in,n), ROR(in,n)
Выбор их двух значений наибольшего, наименьшего	MAX(in0, in1), MIN(in0, in1)
Бинарный выбор	SEL(g, in0, in1)
Мультиплексор	MUX(k, in0,...,inN)

Таблица 1. Основные операторы и функции ST.

Написать ++x; или x += 1; в ST, конечно, нельзя. Единственный вариант – это x := x + 1;.

Полное описание стандартных операций, функций и функциональных блоков вы найдете в руководстве программиста CoDeSys. Развернутые пояснения и примеры даны в [2].

sizeof

В CoDeSys имеется оператор **SIZEOF**(in). Смысл его должен быть очевиден для С программистов.

sprintf

Такого оператора в ST нет. Преобразовать любой переменную любого типа в строку можно с помощью …_TO_STRING.

Указатели.

В МЭК указателей нет, в CoDeSys они есть. Адрес переменной можно получить с помощью оператора ADR, разыменовывание дает '^'. Например:

```
pt : POINTER TO INT;
var_int1:INT;
var_int2:INT;
pt := ADR(var_int1);
var_int2 := pt^;
```

Объединений в CoDeSys V2.3 нет. При необходимости это удается обойти, разместив несколько переменных разного типа по одному адресу в прямо адресуемой памяти либо посредством указателей. Аналогично можно получить доступ к элементам строки 'совместив' ее с массивом.

Точка с запятой.

Не смотря на явное родство ST с языком Паскаль, точка с запятой используется в нем также как в языке С. То есть она служит не разделителем, а признаком конца оператора. Естественно, в одной строке может быть любое число операторов. Кроме того, точка с запятой служит пустым оператором. Текст любого программного компонента (POU) на ST должен включать хотя бы один оператор.

Программные скобки.

В ST нет привычных программных скобок (*{}* в С или *begin* и *end* в Паскале). Вместо этого каждая программная конструкция имеет собственную закрывающую программную скобку. Например: END_FUNCTION или END_IF. Каждому кто имел 'счастье' видеть многоэтажные лесенки завершающих скобок в С, такое решение должно быть понятно. Некоторая многословность текста компенсируется средствами быстрого ввода CoDeSys.

IF ELSE

Оператор выбора IF почти эквивалентен соответствующему оператору в С. Условием может служить переменная или выражение только логического типа. В ином случае, необходимо выполнить явное приведение к логическому типу.

Дополнительно в IF можно вложить последовательную цепочку условий ELSIF. Это достаточно удобно при анализе сложных разветвляющихся условий. Подробнее см. [2].

CASE

Аналогичен switch в С. Альтернативные ветки не имеют закрывающей программной скобки и не могут выполняться одна за другой. Соответственно оператор break не имеет смысла и отсутствует. При указании значений констант альтернативы можно перечислить их через запятую или указать диапазон. Например:

```
CASE (x + 2)/3 OF
0:
      y := 1;
1,2:
      y := 4;
      z := 5;
3..50:
      y := 7
ELSE
      y := 0;
END_CASE
```

WHILE и REPEAT

Цикл WHILE совершенно аналогичен циклу while в C. REPEAT UNTIL соответствует do while. Тонкое, отличие состоит в том, что REPEAT выполнятся до тех пор, пока условие не примет значение TRUE.

FOR

Цикл FOR применяется для выполнения группы выражений заранее известное число раз. Цикл включает установку начального значения, конечное значение, опционально шаг (по умолчанию 1) и тело цикла. Пример:

FOR I := 1 TO 100 BY 10 DO x := x*y; END_FOR

На C этому соответствует:

for (I = 1; I <= 100; I += 10) x = x*y;

Бесконечный цикл в ST обычно записывается как WHILE TRUE DO.

Оператору break в циклах ST соответствует оператор EXIT. Оператор continue в CoDeSys V2.3 отсутствует.

main и PLC_PRG

Прекращение выполнения и возврат управления операционной системе был бы для контроллерной программы полным крахом. Поэтому любая контроллерная программа имеет цикл:

```
while(1)
{
ReadInputs();
PLC_PRG();
WriteOutputs();
}
```

На самом деле в CoDeSys мы видим только PLC_PRG. Сам главный цикл спрятан в системе исполнения. Это наиболее тонкий момент, который нужно очень четко себе представлять. В многозадачных проектах система исполнения создает независимый главный цикл для каждой задачи. Любая ваша программа выполняется в цикле, если вы не предприняли специальных мер. Принимайте это как факт. Сложности в этом нет, устойчивые навыки приобретаются за несколько часов.

Многозадачность.

Многозадачность в CoDeSys доступна даже для ПЛК, выполненных на 8 разрядных микропроцессорах без ОС. В простейшем случае, вместо вызова единственной PLC_PRG, система попеременно подставляет вызовы разных пользовательских программ. Конечно, с учетом их приоритетов и времен циклов выполнения. Так реализуется не вытесняющая многозадачность. В многозадачных проектах необходимо сопоставить программы задачам с помощью менеджера задач в CoDeSys (См. рис. 2). Если этого не сделать, то проект будет однозадачным, с единственной главной функцией PLC_PRG.

Помимо циклических задач в CoDeSys можно создавать задачи, вызываемые по событиям.

В контроллерах имеющих ОС, CoDeSys умеет обеспечивать вытесняющую многозадачность. Например, так работает CoDeSys SP RTE [3]. Минимальное время цикла задач и список активирующих событий необходимо уточнить в документации на контроллер.

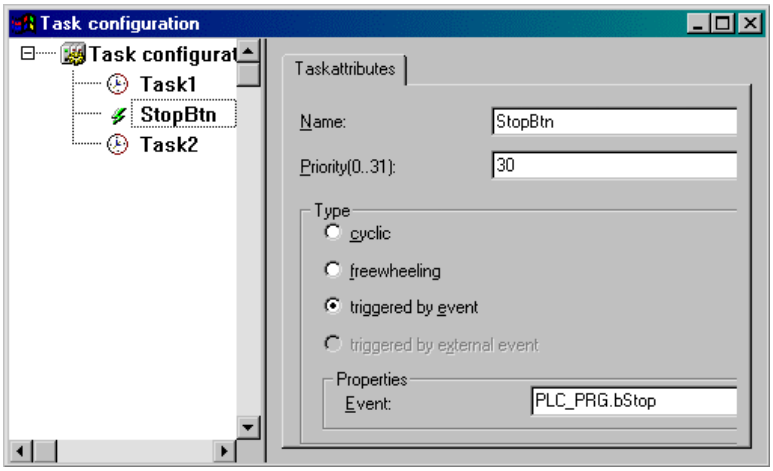

Рис. 2. Настройка задачи, вызываемой по фронту входной переменной.

Функции

Для возврата значения функции в C используется return(). Часто для подготовки возвращаемого значения приходится объявлять вспомогательную переменную. В МЭК каждая функция имеет неявно объявленную переменную для возврата значения. Ее наименование совпадает с именем функции. То есть int GetX(int x) { … return(x); } на ST будет выглядеть так: GetX := x; RETURN;

Как и в языке C передавать параметры, при вызове функции, можно перечисляя их в скобках через запятую: GetX(2). Либо путем присваивания значений: GetX(x := 2). Второй способ применяется и для функциональных блоков.

Память для переменных функции выделяется в стеке. Естественно, значения внутренних переменных не сохраняются между вызовами. Объявления локальных *static* переменных в функциях не предусмотрено. Для функций и функциональных блоков допускается передача параметров по ссылке (VAR_IN_OUT).

Функции МЭК могут иметь переменное число параметров. Типичный пример – это функция MUX, приведенная в таблице 1. Она может работать с переменным числом параметров, причем любого типа. В CoDeSys V2.3 это реализовано только для стандартных функций. Перегрузки стандартных операций нет. Впрочем, этого нет и в языке C.

Типа void в CoDeSys V2.3 нет. Любая функция должна иметь хотя бы один параметр и возвращать значение. Используйте тип BOOL, он обеспечивает минимальные издержки. Как и в C, вызовы функции можно включать в выражения.

Функциональные блоки.

Помимо функции, в МЭК существует понятие функционального блока. По сути, это структура, к которой добавлен управляющий код. При объявлении данных выделяются входные, выходные и внутренние переменные. На основе объявления функционального блока можно объявить один или несколько его экземпляров. Объявление экземпляра приводит к выделению памяти данных, совершенно аналогично созданию переменных типа структура. Естественно код при этом не увеличивается. В отличие от функции, значения переменных экземпляра функционального блока сохраняются между его вызовами. Это означает, что к переменным экземпляра можно обращаться в любое время, вне зависимости от его вызова.

В объявлении экземпляра можно выполнять дополнительную инициализацию переменных. При трансляции с ST в C, для каждого функционального блока приходится писать функцию инициализации.

Дополнительно CoDeSys V2.3 позволяет добавлять действия в функциональные блоки. Это аналогично методам класса в ООП.

Библиотеки.

CoDeSys поддерживает 2 вида библиотек. Первый вид это внутренние библиотеки. Любой проект можно сохранить как библиотеку. Внутренние библиотеки пишутся и хранятся на языках МЭК. Они компилируются вместе с проектом. Вы можете запретить редактирование и просмотр текстов внутренних библиотек пользователем. Специальный менеджер библиотек

позволяет защитить вашу библиотеку лицензией и сделать ее платной. Именно так реализованы библиотеки CANopen и SoftMotion в CoDeSys (язык ST).

Второй тип – это внешние библиотеки. Их код откомпилирован внешними средствами и включен в систему исполнения (например, системные библиотеки) или размещен в obj файлы. Внешние библиотеки могут иметь еще и 'внутреннюю' реализацию (на языках МЭК). Это позволяет написать заглушки системных функций, работающие в режиме эмуляции контроллера в среде программирования CoDeSys.

Системные библиотеки.

Указателей на функции в CoDeSys нет. Есть оператор INDEXOF(name), возвращающий внутренний индекс программного компонента (POU). Он используется там, где нужно программно указать POU. Например, это используется в системных библиотеках, выполняющих установку обработчиков аппаратных прерываний, управление задачами и некоторых других.

В CoDeSys можно создать множество задач и программно управлять (разрешать и останавливать) их работой (по событиям или 'в лоб' с помощью специальной системной библиотеки). Это является аналогом идеи массива указателей на функции.

Динамическое распределение памяти в МЭК не предусмотрено. В CoDeSys это делается с помощью функций специализированной библиотеки. Аналогично поддерживается работа с файлами, потоками, сокетами TCP/IP, точная синхронизация задач, управление работой контроллера и многое другое. Всего в CoDeSys V2.3 существует 24 системные библиотеки.

На самом деле, в CoDeSys видны лишь интерфейсы системных библиотек. Их код реализован в системе исполнения изготовителем. Он решает, что можно отдавать пользователю и что лучше запретить. В итоге, состав системных библиотек может быть разным для разных типов ПЛК. Общий обзор дан в документе SysLyb_Overview_RU в дистрибутиве CoDeSys. Безусловно, использовать системные библиотеки можно порекомендовать только опытным пользователям.

Обработка ошибок.

CoDeSys имеет специальный набор функции с фиксированными именами, позволяющими организовать обработку критических ошибок (деление на ноль, выход за границы массива, диапазоны значений, ошибочные указатели и др.) в пользовательской программе. Например: *Checkbounds, CheckPointer, CheckRangeSigned, CheckRangeUnsigned, CheckDivByte, CheckDivWord, CheckDivDWord* и *CheckDivReal* и др. Если соответствующая функция отсутствует в проекте, то используется обработчик по умолчанию, имеющийся в системе исполнения.

Средства управления проектом.

Все что входит в рабочий проект, CoDeSys сохраняет в единственном файле. Это достаточно удобно для архивации. В случае аварии питания компьютера, CoDeSys полностью восстанавливает проект. В контроллеры, имеющие достаточный объем памяти, можно загрузить файл проекта вместе с исполняемым кодом.

Средств условной трансляции (#ifdef) в CoDeSys V2.3 нет. Однако есть возможность в менеджере проекта выделить определенные POU и запретить их включение в процесс компиляции. В CoDeSys можно установить дополнительный продукт 3S, так называемый инжиниринговый сервер (ENI). С его помощью реализуется полноценная возможность сетевой многопользовательской работы с проектом и управление версиями программных компонентов пользователей.

Заключение.

По числу типов данных ST превосходит C. Применение стандартных и специализированных операций не вызывает у C программистов ни каких сложностей. Редактор ST в CoDeSys поддерживает синтаксическое цветовое выделение и имеет средства быстрого ввода. Как правило, трансляция C программ в ST проще, чем обратная операция. Может вызвать сложность цикличность вызова главной программы, но это дело привычки. Методика программирования и отладки в CoDeSys описана в цикле статей [4].

Вы наверняка заметили, что в определенных местах, где речь шла об ограничениях ST в сравнении с C, было написано 'CoDeSys V2.3'. Все эти моменты доработаны в новой версии CoDeSys V3.0. Безусловно, это радикальный шаг 3S по расширению МЭК. Он сделан под давлением пользователей, являющихся профессиональными программистами. Пожалуй, самое интересное в 3.0 – это оснащение МЭК языков полноценными средствами ООП [5] . На

настоящий момент CoDeSys V3.0 является концептуальной версией и приобретет статус коммерческого продукта к концу 2006 года.

Литература.

1. Пастушенков Д.В., Петров И.В Программируем временные сложности // Промышленные АСУ и контроллеры. 2004, № 7,8,9
2. Петров И.В. Программируемые контроллеры. Стандартные языки и приемы прикладного проектирования / – М.: СОЛОН-Пресс, 2004.
3. Золотарев С.В., Пастушенков Д.В. CoDeSys SP RTE: SoftPLC + возможности реального времени в среде Windows // МКА. 2005. №3
4. Петров И.В. Отладка прикладных ПЛК программ в CoDeSys // Промышленные АСУ и контроллеры. 2006. Начало в № 2.
5. Дитер Хесс, Объектно-ориентированные расширения МЭК 61131-3 // СТА. 2006. № 2.