Работа со строками в CODESYS V3.5

Евгений Кислов, *инженер ОВЕН*

Стандарт МЭК 61131-3 определяет типы данных, которые используются при программировании ПЛК. Они делятся на четыре основных группы: биты, числа, строки и временные типы. В статье описывается работа со строками в среде CODESYS V3.5, применяемой для программирования контроллеров ОВЕН СПК1xx и ПЛК210.

Первые программируемые контроллеры появились в 60-70 годах прошлого века для замены электромеханических реле и аналоговых регуляторов. Тогда для разработки программ было достаточно двух основных типов данных: логического – для представления дискретных сигналов и целочисленного – для представления аналоговых сигналов. Эволюция ПЛК расширила спектр выполняемых задач, что потребовало введения новых типов данных, одним из которых стали строки.

Строки могут использоваться для следующих задач:

- визуализация (формирование таблиц рецептов, сообщений о тревогах);
- запись данных в файлы в понятной человеку форме (в формате CSV, JSON и т. д.);
- реализация строковых протоколов обмена (DCON, MQTT и т. д.);
- работа с SMS;
- хранение паролей, серийных номеров и т.д.

Типы строк в CODESYS V3.5

Строка – это массив чисел, каждое из которых соответствует определенному символу. Соответствие между числами и символами называется кодировкой. В CODESYS V3.5 присутствуют два типа строк – STRING и WSTRING. Характеристики типов строк приведены в табл. 1.

Таблица 1. Характеристики типов STRING и WSTRING

Параметр	STRING	WSTRING
Кодировка	ASCII	UCS-2 (Unicode)
Размер символа	1 байт	2 байта
Пример записи литерала (важен тип кавычек)	'hello, world'	"привет, мир"

Таблица 2. Строковые функции библиотеки Standard

Функция	Краткое описание
CONCAT (STR1, STR2)	Объединяет две строки в одну
DELETE (STR, LEN, POS)	Удаляет из строки заданное число символов с нужной позиции
FIND (STR1, STR2)	Производит поиск подстроки в строке
INSERT (STR1, STR2, POS)	Добавляет подстроку в строку с заданной позиции
LEFT (STR, SIZE)	Выделяет из строки подстроку заданной длины (начиная с первого символа)
LEN (STR)	Вычисляет длину строки
MID (STR, LEN, POS)	Выделяет из строки подстроку заданной длины (начиная с нужной позиции)
REPLACE (STR1, STR2, LEN, POS)	Заменяет в строке один фрагмент на другой (начиная с нужной позиции)
RIGHT (STR, SIZE)	Выделяет из строки подстроку заданной длины (начиная с последнего символа)

Выбор типа зависит от решаемой задачи. Например, для отображения строк в визуализации контроллеров ОВЕН следует использовать только тип WSTRING. При работе с SMS удобнее применять STRING, так как при формировании AT-команд для модемов используется кодировка ASCII.

Длина и размер строки

В CODESYS V3.5 при объявлении строки задается ограничение числа ее символов. Если число символов не указано, то по умолчанию используется значение 80. Диапазон ограничения числа символов строки теоретически бесконечен. Фактически длина строки ограничена только объемом памяти, выделенной под проект.

В CODESYS используются нуль-терминированные строки (как в языке C), то есть каждая строка завершается NUL-символом с кодом «0». Память под этот символ выделяется автоматически, и он не учитывается при объявлении переменной (рис. 1).

Базовые функции работы со строками

Значение строковой переменной можно присвоить не только при ее объявлении, но и в коде программы. Однако одного присваивания недостаточно. Для реализации алгоритмов требуются дополнительные операции, например, объединение нескольких строк в одну, поиск в строке нужного символа и т. д. Для этих операций используются базовые функции из библиотеки Standard. Список этих функций с кратким описанием приведен в табл. 2, примеры использования – на рис. 2.

Функции из библиотеки Standard могут работать только с переменными типа STRING. Для работы с WSTRING

используется библиотека Standard64 с идентичным набором функций, имеющих префикс «W» (WCONCAT, WDELETE и т. д.).

Расширенные функции работы со строками

Важно отметить, что функции из библиотек Standard/Standard64 могут работать только со строками, длина которых не превышает 255 символов. Для работы с более длинными строками используется библиотека StringUtils. В ней содержатся функции, которые в качестве аргументов принимают не строки, а указатели на них. Кроме того, библиотека содержит дополнительные функции для перевода строк в верхний/нижний регистр, удаления пробелов и т. д.

Типы строк STRING и WSTRING предназначены для работы с разными кодировками (табл. 1). Иногда требуется выполнить конвертацию этих типов, например, ввести в визуализацию строку-сообщение типа WSTRING и отправить ее по SMS в виде STRING-значения. Стандартные операторы конверсии STRING_TO_WSTRING/WSTRING_TO_STRING в этом случае не подходят, так как не производят конвертации кодировок, а перекладывают содержимое памяти одной переменной в другую. Решить проблему поможет библиотека OwenStringUtils, разработанная компанией ОВЕН. Библиотека позволяет:

- конвертировать кодировки (рис. 3);
- работать с подстроками;
- форматировать вывод переменных типа DATE/TOD/DT/REAL (рис. 3).

Большой набор функций для работы со строками можно найти в библиотеке OSCAT Basic. Например, зеркалирование строки и преобразование числа в строку с его HEX-значением (рис. 4). Русскоязычное описание библиотеки доступно на сайте owen.ru в разделе CODESYS V3.

Управляющие последовательности

Помимо видимых символов (букв, цифр, знаков препинания) строка может содержать спецсимволы, которые называются управляющими последовательностями. С их помощью, например, можно организовать перевод строки для вывода нескольких сообщений в одном элементе визуализации.

В редакторе CODESYS для ввода спецсимволов используется знак ‘$’ (рис. 5). Полный список спецсимволов приведен в документе CODESYS V3.5. Визуализация.

Строки и массивы

Как было сказано в начале статьи, строка представляет собой массив символов. CODESYS V3.5 позволяет осуществлять индексный доступ к строке – как к массиву значений типа BYTE (для STRING) или WORD (для WSTRING). Это удобно при работе с файлами и реализации протоколов обмена. На рис. 6 приведен пример обработки строки в цикле FOR для определения позиций символов, разделяющих значения. Это может потребоваться при чтении информации из файлов формата .csv.

В некоторых случаях требуется очистить строку. Для этого достаточно присвоить ей «пустое» значение (рис. 7). Но следует учитывать, что эта операция не очищает строку полностью – она только записывает NUL-терминатор в ее начальный символ. На рис. 7 приведен пример, в котором переменной сначала присваивается значение ‘ABCD’, которое потом перезаписывается пустой строкой. Но фактически происходит только обнуление начального символа строки, а коды остальных символов остаются на своих местах. Поэтому записав значение в начальный элемент через индексный доступ, вы получите строку не из одного символа (как могли ожидать), а из четырех. Обычно такие проблемы проявляются при реализации строкового протокола обмена. Чтобы избежать их, надо очищать строку с помощью специальных функций (например, MemFill).

Рассмотрены ключевые моменты работы со строками в CODESYS V3.5. Все перечисленные библиотеки доступны для загрузки на сайте owen.ru в разделе CODESYS V3. Подробная информация о работе со строками приведена в документации к этим библиотекам, а также в справке среды программирования.

```
VAR
    // Максимальная длина – 40 символов
    // Выделенная память – 41 байт
    sMessage: STRING(40) := 'test';

    // Максимальная длина – 80 символов (по умолчанию)
    // Выделенная память – 82 байта
    wsTitle:   WSTRING := "test";
END_VAR
```

Рис. 1

```
    sVar1 := 'Hello, ';
    sVar2 := 'world';

    // sVar3 теперь имеет значение 'Hello, world'
    sVar3 := CONCAT(sVar1, sVar2);

    // iLen будет иметь значение 12
    iLen := LEN(sVar3);
```

Рис. 2

```
    // неправильная конвертация
    // wsMessage получит значение "òâñò"
    wsMessage := TO_WSTRING('тест');

    // правильная конвертация
    // wsMessage получит значение "тест"
    wsMessage := OSU.CP1251_TO_UNICODE('тест');

    // sDateTime получит значение '02.04.2019 08:11:30'
    dtDateTime := DT#2019-04-02-08:11:30;
    sDateTime  := OSU.DT_TO_STRING_FORMAT
                  (dtDateTime, '%t[dd.MM.yyyy HH:mm:ss]');
```

Рис. 3

```
    // sMessage получит значение 'dcba'
    sMessage := BASIC.MIRROR('abcd');

    // sMessage получит значение 'FF'
    sMessage := BASIC.BYTE_TO_STRH(255);
```

Рис. 4

```
    sMessage := 'Один$r$nДва';
```

Рис. 5

Один
Два

```
VAR
            sRecord:          STRING := '123;456;789';
            sSeparatorChar:   STRING := ';';
            aiSeparatorPos:   ARRAY [0..10] OF INT;
            i:                INT;
            j:                INT;
END_VAR

j := 0;
FOR i:= 0 TO LEN(sRecord) DO
            IF sRecord[i] = sSeparatorChar[0] THEN
                        aiSeparatorPos[j] := i;
                        j := j + 1;
// TODO: добавить проверку для верхней границы массива
            END_IF
END_FOR
```

Рис. 6

```
    sMessage := 'ABCD';
    sMessage := '';
    // sMessage получит значение 'EBCD'
    sMessage[0] := 16#45;
```

Рис. 7