Представление данных

CoDeSys

Типы данных

Тип данных определяет род информации, методы ее обработки и хранения, а также количество выделяемой памяти.

Типы данных

Возможно непосредственное использование базовых типов данных и создание пользовательских типов на их основе.

Типы данных

К элементарным (базовым) типам данных относятся:

- Логический (BOOL);
- Целочисленные;
- Рациональные;
- Строки;
- Время и дата.

Типы данных

К пользовательским типам данных относятся:

- Массивы;
- Указатели;
- Перечисление;
- Структуры;
- Псевдонимы типов.

Логический (BOOL)

BOOL - логический тип данных, может принимать одно из двух значений: ИСТИНА (TRUE) или ЛОЖЬ (FALSE).

Логический (BOOL)

Если для объекта типа BOOL не задан прямой битовый адрес, то в памяти выделяется 8 бит.

Логический (BOOL)

Если для объекта типа BOOL не задан прямой битовый адрес, то в памяти выделяется 8 бит.

Целочисленные

К целочисленным типам данным относятся: BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT.

Целочисленные

Тип	Нижний предел	Верхний предел	Размер памяти
BYTE	0	255	8 Бит
WORD	0	65535	16 Бит
DWORD	0	4294967295	32 Бит
SINT	-128	127	8 Бит
USINT	0	255	8 Бит
INT	-32768	32767	16 Бит
UINT	0	65535	16 Бит
DINT	-2147483648	2147483647	32 Бит
UDINT	0	4294967295	32 Бит

Целочисленные

Присвоение данных большего типа переменной меньшего типа может приводить к потере информации.

Рациональные

REAL и LREAL - данные в формате с плавающей запятой, используются для сохранения рациональных чисел. Для типа REAL необходимо 32 бита памяти и 64 для LREAL.

Рациональные

Диапазон значений REAL

от1.175494351e-38

до 3.402823466e+38

Рациональные

Диапазон значений LREAL

от 2.2250738585072014e-308

до 1.7976931348623158e+308

Рациональные

Диапазон значений LREAL

от 2.2250738585072014e-308

до 1.7976931348623158e+308

Строки

STRING - строковый тип представляет собой строку символов.

Строки

Максимальный размер строки определяет количество резервируемой памяти и указывается при объявлении переменной строкового типа.

Строки

Размер задается в круглых или квадратных скобках.

Если размер не указан, принимается размер по умолчанию - 80 символов.

Строки

Длина строки не ограничена в CoDeSys, но строковые функции способны обращаться со строками от 1 до 255 символов!

Строки

Пример объявления строки размером до 35 символов:

str:STRING(35):='Просто строка';

Строки

Пример объявления строки размером до 35 символов:

str:STRING(35):='Просто строка';

Время и дата

TIME представляет длительность интервалов времени в миллисекундах.

Максимальное значение для типа TIME : 49d17h2m47s295ms (4194967295 ms).

Время и дата

TIME_OF_DAY (сокр. TOD) содержит время суток, начиная с 0 часов (с точностью до миллисекунд).

Диапазон значений TOD от:
00:00:00 до 23:59:59.999.

Время и дата

DATE содержит календарную дату, начиная с 1 января 1970 года.

Диапазон значений от:
1970-00-00 до 2106-02-06

Время и дата

DATE_AND_TIME (сокр. DT) содержит время в секундах, начиная с 0 часов 1 января 1970 года.

Диапазон значений от: 1970-00-00-00:00:00 до 2106-02-06-06:28:15.

Время и дата

Типы **TIME, TOD**, **DATE** и **DATE_AND_TIME** (сокр. DT) сохраняются физически как DWORD.

Время и дата

Типы **TIME, TOD**, **DATE** и **DATE_AND_TIME** (сокр. DT) сохраняются физически как DWORD.

Массивы

Элементарные типы данных могут образовывать одно-, двух-, и трехмерные массивы.

Массивы

Массивы могут быть объявлены в разделе объявлений POU или в списке глобальных переменных.

Массивы

Путем вложения массивов можно получить многомерные массивы, но не более 9 мерных ("ARRAY[0..2] OF ARRAY[0..3] OF …").

Массивы

Синтаксис (запись производится в одну строку):

<Имя_массива>:ARRAY
[<ll1>..<ul1>,<ll2>..<ul2>]
OF <базовый тип>;

Массивы

где ll1, ll2, ll3 указывают нижний предел индексов; ul1, ul2 и ul3 указывают верхние пределы.

Массивы

Индексы должны быть целого типа. Нельзя использовать отрицательные индексы.

Массивы

Пример:

Card_game: ARRAY [1..13, 1..4] OF INT;

Массивы

Пример инициализации простых массивов:

```
arr1 : ARRAY [1..5] OF INT := 1,2,3,4,5;
arr2 : ARRAY [1..2,3..4] OF INT := 1,3(7); (*
сокращение для 1,7,7,7 *)
arr3 : ARRAY [1..2,2..3,3..4] OF INT :=
2(0),4(4),2,3;
      (* сокращение для 0,0,4,4,4,4,2,3 *)
```

Массивы

Пример инициализации массива структур:

```
TYPE STRUCT1
STRUCT
p1:int;
p2:int;
p3:dword;
END_STRUCT
```

Массивы

Пример инициализации массива структур:

ARRAY[1..3] OF STRUCT1:=
(p1:=1,p2:=10,p3:=4723),(p1:=2,p2:=0,p3:=
299),
(p1:=14,p2:=5,p3:=112);

Массивы

Пример инициализации части массива:

```
arr1 : ARRAY [1..10] OF INT := 1,2;
```

Массивы

Не инициализированные явно элементы массива принимают значения по умолчанию.

Так, в данном примере оставшиеся элементы примут значение 0.

Массивы

Доступ к элементам массива:

Для доступа к элементам двухмерного массива используется следующий синтаксис:

<Имя_массива>[Индекс1,Индекс2]

Массивы

Пример:

Card_game [9,2]

Функция CheckBounds

Определив в проекте функцию с именем **CheckBounds,** возможно использовать её для контроля за соблюдением границ индексов массивов.

Функция CheckBounds

Имя функции фиксировано, изменять его нельзя.

Функция CheckBounds

Пример функции CheckBounds:

```
FUNCTION CheckBounds : INT
VAR_INPUT
 index, lower, upper: INT;
END_VAR
IF  index < lower THEN
 CheckBounds := lower;
ELSIF  index > upper THEN
 CheckBounds := upper;
ELSE  CheckBounds := index;
END_IF
```

Функция CheckBounds

В этом примере CheckBounds ограничивает индекс массива заданными границами. Если запрашивается элемент, отсутствующий в массиве, функция CheckBounds возвращает ближайший элемент.

Функция CheckBounds

Функция CheckBounds, содержащаяся в библиотеке Check.Lib, представляет собой пример реализации.

Указатели

Указатели позволяют работать с адресами переменных или функциональных блоков.

Указатели

Синтаксис:

<Имя_указателя>: **POINTER TO** <Тип данных/Функциональный блок>;

Указатели

Указатели применимы для всех базовых типов данных или функциональных блоков, включая определяемые пользователем.

Указатели

Адреса переменных и функциональных блоков можно получить во время исполнения программы при помощи оператора ADR.

Указатели

Для обращения через указатель необходимо добавить оператор "^" (content) после его имени.

Указатели

Указатели инкрементируются побайтно! Для увеличения указателя, как это принято в С-компиляторах, используйте инструкцию

p=p+SIZEOF(p^);

Указатели

Пример:

```
pt:POINTER TO INT;
var_int1:INT := 5;
var_int2:INT;
pt := ADR(var_int1);
var_int2:= pt^;  (* var_int2 теперь равна 5 *)
```

Функция CheckPointe

Данная функция позволяет контролировать обращение к допустимой области памяти через указатели. Если определена функция CheckPointer, то она будет автоматически вызываться при любом обращении через указатель.

Функция CheckPointe

Функция должна быть определена в проекте (непосредственно или в библиотеке). Ее имя (CheckPointer) изменять нельзя.

Функция CheckPointe

Функция возвращает адрес, который будет использоваться как указатель.

Перечисление

Перечисление - это определяемый пользователем тип данных, задающий несколько строковых псевдонимов для числовых констант.

Перечисление

Перечисление доступно в любой части проекта, даже при локальном его объявлении внутри POU.

Перечисление

Поэтому наиболее разумно создавать все перечисления на вкладке «Типы данных» Организатора Объектов.

Перечисление

Объявление должно начинаться с ключевого слова TYPE и заканчиваться строкой END_TYPE.

Перечисление

Синтаксис:

TYPE <Имя_перечисления>:(<Элемент_0> ,< Элемент _1>, ...< Элемент _n>);END_TYPE

Перечисление

Переменная типа <Имя_перечисления> может принимать только перечисленные значения.

Перечисление

При инициализации переменная получает первое из списка значение.

Перечисление

Если числовые значения элементов перечисления не указаны явно, им присваиваются последовательно возрастающие числа, начиная с 0.

Перечисление

Фактически элемент перечисления - это число типа INT и работать с ними можно точно так же. Можно напрямую присвоить число переменной типа перечисление.

Перечисление

Пример:

TYPE TRAFFIC_SIGNAL: (Red, Yellow, Green:=10); END_TYPE

(*Каждому цвету
соответствует свое значение, для red - это 0, для yellow - 1 и для green - 10 *)

Перечисление

Продолжение пример:

```
TRAFFIC_SIGNAL1 : TRAFFIC_SIGNAL;
TRAFFIC_SIGNAL1:=0;

(* Переменная получила значение red*)
```

Перечисление

Продолжение пример:

```
FOR i:= Red TO Green DO
 i := i + 1;
END_FOR;
```

Перечисление

Элемент, уже включенный в перечисление, нельзя повторно включать в другое перечисление.

Псевдонимы типов

Псевдонимы типов нужны для создания альтернативных пользовательских наименований типов данных. Это удобно при работе с большим числом однотипных констант, переменных и функциональных блоков.

Псевдонимы типов

Псевдонимы типов определены на вкладке **Типы данных** Организатора Объектов. Объявление должно начинаться с ключевого слова TYPE и заканчиваться строкой END_TYPE.

Псевдонимы типов

Синтаксис:

TYPE <Имя псевдонима>: <Исходное имя>;
END_TYPE

Псевдонимы типов

Пример:

```
TYPE message:STRING[50];
END_TYPE;
```