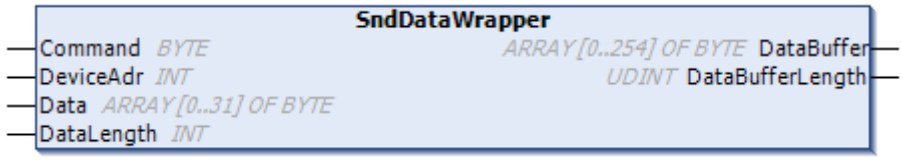
SndDataWrapper (FB)

FB подготавливает данные для отправки в прибор в зависимости от команды. Данные на входе: команда, данные команды, длина данных команды, адрес прибора. Данные на выходе: сформированный массив данных, длина буфера для отправки.

InOut:

Scope	Name	Type	Comment
Input	Command	BYTE	код команды
	DeviceAdr	INT	адрес устройства
	Data	ARRAY [0..127] OF BYTE	данные команды, если применимо
	DataLength	INT	длина данных команды
Output	DataBuffer	ARRAY [0..254] OF BYTE	выходной буфер данных для отправки
	DataBufferLength	UDINT	длина буфера данных для отправки



Графическое представление функционального блока SndDataWrapper

==========

Объявление переменных

==========

```
/// FB подготавливает данные для отправки в прибор в зависимости от команды.
/// Данные на входе: команда, данные команды, длина данных команды, адрес прибора.
/// Данные на выходе: сформированный массив данных, длина буфера для отправки.

{attribute 'hide_all_locals'}
FUNCTION_BLOCK SndDataWrapper
VAR_INPUT
        Command: BYTE;                          (* код команды *)
        DeviceAdr: INT;                         (* адрес устройства *)
        Data: ARRAY [0..127] OF BYTE;           (* данные команды, если применимо *)
        DataLength: INT;                        (* длина данных команды *)
END_VAR
VAR_OUTPUT
        DataBuffer: ARRAY [0..254] OF BYTE;     (* выходной буфер данных для отправки *)
        DataBufferLength: UDINT;                (* длина буфера данных для отправки *)
END_VAR
VAR
        TmpBuff: ARRAY [0..127] OF BYTE;        // промежуточный буфер для работы с данными
        i, j, z: INT := 0;
        SpecSymbCnt: INT := 0;                  // счётчик дополнительных спец. символов
END_VAR
VAR CONSTANT
        Separator: BYTE := 16#FF;               // разделитель
END_VAR
```

==========

Код исполнения

==========

```
(*
Структура посылки:
 <FF><Adr><COP><Data><CRC><FF><FF>
 <FF> - разделитель
 <Adr> - адрес устройства
 <COP> - код операции
```

```
 <Data> - данные (используются не во всех командах)
 <CRC> - CRC
*)

// Формирование буфера данных
DataBuffer[0] := Separator; // сепаратор
DataBuffer[1] := INT_TO_BYTE(DeviceAdr); // адрес устройства
DataBuffer[2] := Command; // код команды

// Заполняем буфер массивом данных <Data>, которые необходимо передать
IF DataLength <> 0 THEN
	FOR i := 0 TO (DataLength - 1) DO
		DataBuffer[i+3] := Data[i];
	END_FOR
END_IF

// Очистка буфера данных для расчёта CRC
(*
Функция MemFill в Codesys заполняет блок памяти заданной длины заданными значениями.
ADR(TmpBuff) – указатель на стартовый байт блока памяти, который будет заполнен.
128– длина блока памяти для заполнения, в байтах.
16#0 – значение, которым будет заполнен блок памяти.

В других средах разработки должна быть использована другая инструкция, которая заполнит массив TempBuff нулевыми значениями.
*)
MEM.MemFill(ADR(TmpBuff), 128, 16#0);

// Заполняем промежуточный массив, который будет использоваться для расчёта CRC
FOR i := 0 TO (DataLength+1) DO
	TmpBuff[i] := DataBuffer[i+1];
END_FOR

// Добавляем в массив данных CRC
DataBuffer[3+DataLength] := CRC_Calc(ADR(TmpBuff[0]), 2+DataLength);

(*
Дополнительная проверка на наличие в области <Data><CRC> байта 16#FF (область <Adr><COP> не проверяем, т.к. они не должны принимать эти значения по спецификации протокола).
Если встречается байт 16#FF, то после него вставляем 16#FE.
Проверку реализуем на этом этапе, т.к. вставленные байты 16#FE не должны участвовать в расчёте CRC.
*)
z := 0; // начальная позиция заполнения временного буфера данных
SpecSymbCnt := 0; // обнуляем счётчик доп. спец. символов
FOR j := 3 TO (DataLength + 3) DO
	TmpBuff[z] := DataBuffer[j];
	z := z + 1;
	IF DataBuffer[j] = 16#FF THEN
		TmpBuff[z] := 16#FE;
		z := z + 1;
		SpecSymbCnt := SpecSymbCnt + 1;
	END_IF
END_FOR
(*
Функция SysMemCpy в Codesys используется для копирования заданного кол-ва байтов из одного блока памяти в другой.
ADR(DataBuffer[3]) – указатель на стартовый байт блока памяти, в который будут скопированы данные.
ADR(TmpBuff[0]) – указатель на стартовый байт блока памяти, из которого будут скопированы данные.
128 – длина блока памяти, который будет скопирован.
```

```
*В других средах разработки должна быть использована другая инструкция, которая скопирует 128 элементов массива TmpBuff, начиная с индекса 0, в массивDataBuffer , начиная с индекса 3.
*)
Util.SysMem.SysMemCpy(pDest := ADR(DataBuffer[3]), pSrc := ADR(TmpBuff[0]), udiCount := 128);

// Добавляем сепараторы в конце массива данных
DataBuffer[4+DataLength+SpecSymbCnt] := Separator; // сепаратор
DataBuffer[5+DataLength+SpecSymbCnt] := Separator; // сепаратор

// Расчёт длины буфера данных для отправки
DataBufferLength := TO_UDINT(6 + DataLength + SpecSymbCnt);

================================================================================================
```