RcvDataParser (FB)

FB для проверки полученных данных и удаления служебных битов. На выходе выдаёт чистый блок <Data> без служебных битов. В случае ошибки при проверке данных формирует флаг и тип ошибки.
Критерии проверки полученных данных:

1. Наличие сепараторов в начале и конце блока данных.
2. Проверка адреса устройства.
3. Совпадение команд в запросе и ответе.
4. Проверка CRC.

InOut:

Scope	Name	Type	Initial	Comment
Input	Enable	BOOL		Запуск на исполнение
	InData	ARRAY [0..254] OF BYTE		Массив данных для обработки
	Command	BYTE		Отправленная в устройство команда
	DeviceAdr	INT		Адрес устройства, от которого предполагалось получить данные
Output	Error	BOOL	FALSE	Флаг ошибки
	ErrorType	INT	0	тип ошибки
	OutData	ARRAY [0..254] OF BYTE		массив выходных данных
	Done	BOOL	FALSE	флаг завершения проверки без ошибок

Графическое представление функционального блока RcvDataParser

==========

Объявление переменных

==========

```
/// FB для проверки полученных данных и удаления служебных битов.
/// На выходе выдаёт чистый блок <Data> без служебных битов. В случае ошибки при проверке данных формирует флаг и тип ошибки.

{attribute 'hide_all_locals'}
FUNCTION_BLOCK RcvDataParser
VAR_INPUT
	Enable: BOOL;			(* Запуск на исполнение *)
	InData: ARRAY [0..254] OF BYTE;	(* Массив данных для обработки *)
	Command: BYTE;			(* Отправленная в устройство команда *)
	DeviceAdr: INT;			(* Адрес устройства, от которого предполагалось получить данные *)
END_VAR
VAR_OUTPUT
	Error: BOOL := FALSE;		(* Флаг ошибки *)
	ErrorType: INT := 0;		(* тип ошибки *)
	OutData: ARRAY [0..254] OF BYTE;	(* массив выходных данных *)
	Done: BOOL := FALSE;		(* флаг завершения проверки без ошибок *)
END_VAR
VAR
	i, j, z: INT := 0;
	TempBuff: ARRAY [0..254] OF BYTE;	// Внутренний буфер для работы с данными
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
*)
IF Enable THEN
	(*
	Проверка данных.
	Критерии проверки:
		1. Наличие сепараторов в начале и конце блока данных.
		2. Проверка адреса устройства.
		3. Совпадение команд в запросе и ответе.
		4. Проверка CRC.
	*)

	// Сброс флагов и типа ошибки
	Error := FALSE;
	ErrorType := 0;
	Done := FALSE;

	// Проверка на наличие символа-разделителя в начале блока данных
	IF InData[0] <> 16#FF THEN
		Error := TRUE;
		ErrorType := 5061;
		JMP _lbl_1; (* при возникновении ошибки оставшиеся проверки не производим *)
	END_IF

	// Проверка адреса устройства
	IF InData[1] <> INT_TO_BYTE(DeviceAdr) THEN
		Error := TRUE;
		ErrorType := 5062;
		JMP _lbl_1;  (* при возникновении ошибки оставшиеся проверки не производим *)
	END_IF

	// Проверка на совпадение команд в запросе и ответе, 16#EE и 16#FD не приводят к ошибке
	(*
	16#FD – указывает на ответ на неподдерживаемую команду, которую использовали для получения информации о приборе, но при необходимости можно реализовать ошибку по данному ответу.
	16#EE – указывает на ответ, содержащий ошибку от прибора. Обработка этих ошибок была реализована в другом FB. При необходимости можно реализовать здесь.
	*)
	IF InData[2] <> Command AND InData[2] <> 16#EE AND InData[2] <> 16#FD THEN
		Error := TRUE;
		ErrorType := 5063;
		JMP _lbl_1; (* при возникновении ошибки оставшиеся проверки не производим *)
	END_IF

	// Проверка на наличие символов-разделителей в конце блока данных
	(* должны получить подряд два байта 16#FF *)
	Error := TRUE; // взводим флаг ошибки, будет сброшен, если разделители на месте
	FOR i := 3 TO 253 DO
		IF InData[i] = 16#FF AND InData[i+1] = 16#FF THEN
```

```
                    Error := FALSE;
                    EXIT; // выход из цикла FOR, в других средах может быть другой оператор, например BREAK
            END_IF
END_FOR
IF Error THEN
            ErrorType := 5064;
            JMP _lbl_1; (* при возникновении ошибки оставшиеся проверки не производим *)
END_IF

// Проверка CRC
(*
Функция MemFill в Codesys заполняет блок памяти заданной длины заданными значениями.
ADR(TempBuff) – указатель на стартовый байт блока памяти, который будет заполнен.
255 – 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(TempBuff), 255, 16#0); // очистка буфера данных для расчёта CRC
// Копируем во внутренний буфер <Adr> и <COP>
(*
Функция SysMemCpy в Codesys используется для копирования заданного кол-ва байтов из одного блока памяти в другой.
ADR(TempBuff[0]) - указатель на стартовый байт блока памяти, в который будут скопированы данные.
 ADR(InData[1]) – указатель на стартовый байт блока памяти, из которого будут скопированы данные.
2 – длина блока памяти, который будет скопирован.

 В других средах разработки должна быть использована другая инструкция, которая скопирует 2 элемента массива InData, начиная с индекса 1, в массив TempBuff , начиная с индекса 0.
*)
Util.SysMem.SysMemCpy(pDest := ADR(TempBuff[0]), pSrc := ADR(InData[1]), udiCount := 2);
// Отбрасываем дополнительные байты 16#FE из блока данных, если они есть и следуют за байтом 16#FF.
j := 3; // стартовый индекс массива, который проверяем
z := 2; // стартовый индекс массива, в который складываем данные
WHILE j <= i-1 DO
            TempBuff[z] := InData[j];
            z := z + 1;
            IF InData[j] = 16#FF AND InData[j+1] = 16#FE THEN
                        j := j + 2;
            ELSE
                        j := j + 1;
            END_IF
END_WHILE
// Рассчитываем CRC вместе с полученным, должны получить 0
IF CRC_Calc(ADR(TempBuff[0]), z) <> 0 THEN
            Error := TRUE;
            ErrorType := 5065;
            JMP _lbl_1;
END_IF

// Если ошибок не выявлено, то взводим флаг завершения
Done := TRUE;

_lbl_1: ;

// Выдача чистого блока <Data> на выход FB
IF NOT Error THEN
            // Удаляем CRC из буфера, чтобы не выдавать его на выход
            TempBuff[z-1] := 16#00;
            (* Используем массив TempBuff, т.к. он содержит все необходимые данные с отброшенными служебными символами *)
```

```
                (* В другой среде разработки функцию SysMemCpy необходимо заменить по аналогии с предыдущим описанием *)
                Util.SysMem.SysMemCpy(pDest := ADR(OutData[0]), pSrc := ADR(TempBuff[2]), udiCount := 253);
        ELSE
                // В случае наличия ошибки на выходе нулевые данные
                (* В другой среде разработки функцию MemFill необходимо заменить по аналогии с предыдущим описанием *)
                MEM.MemFill(ADR(OutData), 255, 16#0);
        END_IF
ELSE
        Done := FALSE; // снимаем флаг при снятии Enable
END_IF

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