

НАСТРОЙКА ОБМЕНА ДАННЫМИ ПО ПРОТОКОЛУ НАRT НА КОНТРОЛЛЕРАХ СЕРИИ REGUL RX00

Руководство пользователя

DPA-302.3 Версия документа 1.10 Версия ПО 1.7.1.0 Август 2023

Версия руководства пользователя	Описание изменения
1.6	Добавлена история изменений руководства пользователя.
	Добавлены знаки с предупреждающей и поясняющей информацией.
	Раздел «Алгоритм работы HART-мастера»: дополнено описание режима работы в зависимости от типа модуля.
	Дополнительно по тексту внесены небольшие изменения с уточняющей информацией
1.7	Добавлен новый раздел: «Поддержка устройств в Burst режиме».
	Дополнительно по тексту внесены небольшие изменения с уточняющей информацией
1.8	Выпуск среды разработки Astra.IDE.
	Подраздел «Перечень действий при настройке HART»: добавлена информация об исключении функции подключения HART-устройств к последовательному порту, начиная с версии СПО 1.7.0.0.
	Подраздел «Добавление HART-устройств к модулям аналогового ввода/вывода»: добавлена информация об ограничении максимального количества HART-мастеров в проекте
1.9	По тексту внесены небольшие изменения с уточняющей информацией
1.10	Удалены следующие подразделы:
	– «Добавление HART-устройств к коммуникационным модулям»;
	 «Добавление HART-устройств непосредственно к модулю центрального процессора».
	Подраздел «Добавление HART-устройств к модулям аналогового ввода/вывода»: изменено значение максимального количества HART-мастеров в проекте.
	Подраздел «Привязка переменных программы к командам» Сбрасывать переменные для привязки при обновлении HartDevice в редакторе Hart_Outer_Slave ⇔ Hart команды

История изменений руководства пользователя

⇒АННОТАЦИЯ

Настоящий документ содержит сведения о настройке передачи данных с применением протокола HART на промышленных логических контроллерах серии Regul RX00. Настройка осуществляется с помощью программного обеспечения Astra.IDE.

Данное руководство предназначено для эксплуатационного персонала и инженеровпроектировщиков АСУ ТП, которые должны:

- иметь, как минимум, среднее техническое образование;
- приступить к работе только после изучения данного руководства.

Обновление информации в Руководстве

Производитель ООО «РегЛаб» оставляет за собой право изменять информацию в настоящем Руководстве и обязуется публиковать более новые версии с внесенными изменениями. Обновленная версия Руководства доступна для скачивания на официальном сайте Производителя: https://reglab.ru/.

Для своевременного отслеживания выхода новой версии Руководства рекомендуется оформить подписку на обновление документа. Для этого необходимо на сайте Производителя: https://reglab.ru/ кликнуть на кнопку «Подписаться на обновления» и оставить свои контактные данные.

В руководстве присутствуют знаки с предупреждающей и поясняющей информацией. Каждый знак обозначает следующее:

ПРЕДУПРЕЖДАЮЩИЕ ЗНАКИ



ВНИМАНИЕ!

Здесь следует обратить внимание на способы и приемы, которые необходимо в точности выполнять во избежание ошибок при эксплуатации или настройке.

ИНФОРМАЦИОННЫЕ ЗНАКИ



ИНФОРМАЦИЯ

Здесь следует обратить внимание на важную информацию

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	5
Общие сведения	5
Перечень рекомендуемых документов	5
НАСТРОЙКА РАБОТЫ	6
Перечень действий при настройке HART	6
Алгоритм работы HART-мастера	7
ДОБАВЛЕНИЕ И НАСТРОЙКА УСТРОЙСТВ, РАБОТАЮЩИХ ПО	
ПРОТОКОЛУ HART	9
Добавление устройств и объектов в конфигурацию контроллера	9
Добавление HART-устройств к модулям аналогового ввода/вывода	9
ДОБАВЛЕНИЕ ПОЛЬЗОВАТЕЛЬСКИХ КОМАНД, ПЕРЕДАВАЕМЫХ	ПО
HART	13
Добавление контейнеров HartDevice	13
Использование предустановленного набора команд	16
Дизайн собственных команд	18
Редактор hart команды	18
Библиотека PsIoDrvHartMaster	21
ОБЪЯВЛЕНИЕ ПЕРЕМЕННЫХ И ПРИВЯЗКА ПЕРЕМЕННЫХ К	
КОМАНДАМ	23
Объявление переменных	23
Привязка переменных программы к командам	23
ОБРАБОТКА ВЫПОЛНЕНИЯ КОМАНДЫ	27
ПОДДЕРЖКА УСТРОЙСТВ В BURST РЕЖИМЕ	29
ФУНКЦИОНАЛЬНЫЙ БЛОК HARTUSERREQUEST И ДИНАМИЧЕСН	٢И
ФОРМИРУЕМАЯ КОМАНДА	31
Общее описание	31
Объявление структуры данных команды непосредственно в коде	32
Использование команды из контейнера команд HART-устройства	34
Пример выполнения команды 34 – Write Primary Variable Damping Value	35

введение

Общие сведения

Программное обеспечение контроллера позволяет сконфигурировать его в качестве HART-мастера (по умолчанию – primary) и опрашивать Slave-устройства (интеллектуальные датчики), либо управлять ими (исполнительные устройства с поддержкой HART) по последовательной линии по протоколу HART 6.

Протокол HART основан на методе передачи данных с помощью частотной модуляции (Frequency Shift Keying, FSK), в соответствии с коммуникационным стандартом Bell 202. Цифровая информация передаётся частотами 1200 Гц (логическая 1) и 2200 Гц (логический 0), которые накладываются на аналоговый токовый сигнал. Скорость передачи данных для HART составляет 1,2 кбит/с.

Работа осуществляется в двух режимах:

- Одноточечный режим соединение «точка-точка», при этом протокол допускает параллельную работу двух мастеров (например, первичный-стационарный контроллер и вторичный- мобильный ручной коммуникатор);
- Многоточечный режим объединение в сеть нескольких ведомых устройств и двух мастеров. При этом по линии осуществляется только цифровая связь. Только одно из ведомых устройств на шине может работать в особом режиме ускоренной передачи (burst mode), в котором оно периодически отправляет в сеть пакет – ответ на заданную команду.

Начиная с версии СПО 1.6.5.0, доступна нативная версия драйвера Hart Master OS с поддержкой протокола HART 7 и технологии FDT/DTM. Подробное описание подключения к контроллеру по спецификации DTM приведено в документе «Настройка и работы REGUL GW DTM. Руководство пользователя».

Перечень рекомендуемых документов

Для получения информации по настройке других параметров контроллеров серии Regul RX00 в среде разработки Astra.IDE рекомендуется ознакомиться со следующими документами (доступны на сайте https://reglab.ru/):

- Программное обеспечение Astra.IDE. Руководство пользователя;
- Regul R600. Системное руководство;
- Regul R500. Системное руководство;
- Regul R200. Системное руководство.

НАСТРОЙКА РАБОТЫ

Установите на компьютер программное обеспечение Astra.IDE. Описание процесса установки программы, а также инструкции по работе с программой приведены в документе «Программное обеспечение Astra.IDE. Руководство пользователя». Программа установки и документация доступны на сайте www.prosoftsystems.ru.

Запустите программу Astra.IDE. Откройте проект, в котором требуется настроить контроллер для обмена данными по протоколу HART. Если такого проекта нет, создайте его с помощью Мастера конфигурации Regul (описание приведено в разделе «Основные понятия среды разработки. Проект» документа «Программное обеспечение Astra.IDE. Руководство пользователя DPA-302»).

Перечень действий при настройке HART

Настройка НАRT начинается с добавления в проект НАRT-мастера (Hart Master), к которому, в свою очередь, должны быть добавлены конечные устройства Hart Outer Slave, непосредственно осуществляющие обмен данными по HART-протоколу. При использовании в проекте модулей ввода/вывода с поддержкой HART (см. соответствующее описание модуля в «Regul RX00. Системное руководство») устройство HART-мастер (Hart Master) добавляется непосредственно к модулю ввода/вывода (см. раздел «Добавление НАRT-устройств к модулю аналогового ввода/вывода») и ассоциируется с выбранным каналом этого модуля, а каждый экземпляр Hart Outer Slave на этом канале обозначает отдельное устройство (датчик) в сети.

Далее необходимо создать в проекте объекты типа **HartDevice**. Это контейнеры, содержащие HART-команды для slave-устройства. Следующий шаг – добавление команд в контейнер (см. раздел «Добавление пользовательских команд, передаваемых по HART»).

В МЭК-приложении следует объявить переменные соответствующего типа для каждой команды в контейнере HartDevice. Далее необходимо выполнить привязку контейнера с командами к slave-устройству и привязку объявленной переменной к параметрам команды (см. раздел «Объявление переменных и привязка переменных к командам»).

Для выполнения HART-команды в МЭК-приложении организуется цикл обработки с анализом статуса команды. И, при переходе команды из фазы выполнения (InProcess) в состояние завершения команды (с ошибкой или без), производится обработка полученных данных. Значение статуса *Ok* устанавливается при успешном получении ответа от устройства и его обработке без ошибок (см. раздел «Обработка выполнения команды»).

Алгоритм работы HART-мастера

Одноточечный режим

Если на один вход/выход аналогового модуля с поддержкой HART подключено одно HART-устройство, то одновременно доступно получение как самого токового значения с датчика (его Primary Value, PV), так и обмен данными с устройством по HART-протоколу. При этом в устройстве, согласно спецификации HART (например, с помощью команды 6), рекомендуется задать адрес 0 и включить режим использования токового сигнала (заводские настройки по умолчанию).

При этом протокол допускает параллельную работу двух мастеров:

- первичное ведущее устройство(Primary);
- вторичное ведущее устройство (Secondary).

Многоточечный режим

Если на один вход/выход подключено несколько HART-устройств, то им назначаются уникальные адреса на шине и токовый сигнал переводится в минимально необходимое для функционирования устройства значение (4 мА). При этом работает только цифровая часть HART-протокола — обмен командами. Дополнительно, на уровне Hart_Outer_Slave осуществляется синхронизация доступа к одному каналу модуля — после захвата и выполнения одной попытки запроса (успешной или с ошибкой) объект Hart_Outer_Slave гарантированно освобождает канал на заданный период времени (100 мс).

ИНФОРМАЦИЯ

Определенные модули ввода/вывода контроллеров серии Regul RX00, оборудованные отдельным HART-модемом на каждый канал, позволяют подключать на выбранный канал до 10 HART-устройств.

В случае, если в модуле один HART-модем с помощью мультиплексирования обслуживает группу каналов, то, для оптимизации обмена данными, рекомендуется подключать на один канал модуля не более одного HART-устройства

Согласно спецификации HART поддерживается арбитраж шины со вторым мастером и поддержка устройств, работающих в Burst режиме.

Режим работы каналов модуля

Если каждый канал модуля работает через собственный HART-модем, то выполнение обмена данными по протоколу HART по всем каналам происходит параллельно и независимо от соседнего канала. При этом обработка всех подключенных на один канал устройств с поддержкой HART производится последовательно, по одной команде, с синхронизацией доступа к HART-модему.

Если же каналы в модуле разделены на группы и каждая группа работает через один HART-модем, то производится двухуровневая синхронизация доступа. На первом уровне доступ к HART-модему последовательно пытаются получить экземпляры устройств **Hart Master**, добавленные в дерево устройств к соответствующим каналам модуля в рамках одной подгруппы каналов (например, каналы 1-8 и 9-16 в модуле R500 AI 16 081). Далее, на втором уровне, обработка всех подключенных на один канал устройств с поддержкой HART производится последовательно, по одной команде, также с синхронизацией доступа к HART-модему.



ИНФОРМАЦИЯ

Рекомендуется производить подключение конечных HART-устройств с равномерным распределением по группам каналов модуля и по каналам модуля в рамках одной подгруппы

ДОБАВЛЕНИЕ И НАСТРОЙКА УСТРОЙСТВ, РАБОТАЮЩИХ ПО ПРОТОКОЛУ HART

Добавление устройств и объектов в конфигурацию контроллера

Общий принцип добавления устройств/объектов в конфигурацию контроллера описан в разделе «Описание интерфейса. Добавление объектов» документа «Программное обеспечение Astra.IDE. Руководство пользователя». Далее добавляют соответствующие HART-устройства (Рисунок 1)



Рисунок 1 – НАКТ-устройства



İ

ВНИМАНИЕ!

Начиная с версии СПО 1.7.0.0, функция подключения HART-устройств к последовательному порту исключена

ИНФОРМАЦИЯ

Начиная с версии СПО 1.6.5.0, при выборе устройства Hart доступна нативная версия драйвера с дополнением OS:

1 Hart Master 1 Hart Master "Prosoft-Systems" Ltd. 1.7.0.0 Устройство, которое работает как Hart мастер через последовательный интерфейс 1 Hart Master OS "Prosoft-Systems" Ltd. 1.7.0.0 Устройство, которое работает как Hart мастер через последовательный интерфейс. Драйвер на уровне ОС.

Добавление HART-устройств к модулям аналогового ввода/вывода

Добавьте в проект модуль аналогового ввода/вывода с поддержкой HART. К модулю можно добавить один или несколько HART-мастеров (Hart Master), каждый из которых является контейнером для нескольких устройств Hart Outer Slave (Рисунок 2). <u>Максимальное</u>

количество HART-мастеров в проекте – 1000, но с учетом общего ограничения наложенного на конфигурацию контроллера.

Regul_Bus (Regul_Bus (I Bus) (naŭt)	Î	Добавить устройство				ß					
	<1x (S	(pen) T xx x1x) 19.00 xxx)		Имя: Hart_Master									
*: CU_00_xxx (CU 00 xxx)				Действие									
				🛞 Добавить устройство 💿 Вставить устройство 💿 Подключять устройство 💿 Обновить устройство									
∽ ∦ st_x_	È	Копировать											
		Вставить		Строка для полнотекстового поиска	ndors>								
	×	Удалить		Имя	Поста	вщик	Версия	Описание:					
		Рефакторинг •		🖃 📆 Regul									
	6	Свойства		Hart									
		Добавление объекта		Hart Master	"Proso	ft-Systems" Ltd.	1.7.0.0	Устройство, которое работает как Hart мастер на 16 канальных молулях					
	-0	Добавить папку		19 Hart Master OS	"Proso	ft-Systems" Ltd.	1.7.0.*	Устройство, которое работает как Hart мастер на 16 канальных модулях. Драйвер на уровне ОС					
		Добавить устройство											
		Вставить устройство											
		Откл. устройство											
		Обновить устройство											
				Группировать по категориям	Отобра	жать все версии (для эксперт	ов) 🔲 Показать устаревшие версии					
				1 Имя: Hart Master	atomo" Li	d							
				Группы: Hart Master	stems Li	u.							
				Версия: 1.7.0.0 Номер молели: -									
				Описание: Устройство, котор	юе рабо	тает как Hart маст	гер на 16 ка	нальных модулях					
				R -6									
				АІ 16 081	Kak no	леднего потом	Kd						
				 (Можно выбрать другой таргет 	-узел, п	ока окно открыто	.)						
				L				Добавить устройство Закрыть					

Рисунок 2 – Добавление Hart Master

Далее к устройству **Hart Master** нужно подключить одно или несколько внешних slave-устройств (outer slaves), которые будут опрашиваться контроллером (*Regul* \rightarrow *Hart* \rightarrow *Hart Master* \rightarrow *Hart Outer Slave*) (Рисунок 3).

in 17 Regul_Bus (Regul Bus) in 17 R500_(R500 Κρε∛τ) - 21 ST_XX_X1X (ST XX X1X) - 21 ST_XX_X1X (ST 00 0 XYX)	Добавить устройство Имя: Hart_Outer_Slave			
	Действие Добавить устройство Всгав	ить устройство 🔘 Подкл	ючить устрой	іство 🔘 Обновить устройство
— 17 2 : Нагt 🛛 😽 Вырезать — 17 3 : Нагt I 📔 Копировать	Строка для полнотекстового поиска	Поставщик	<all td="" ven<=""><td>dors> 🔻</td></all>	dors> 🔻
-17 4: Hart_1 Вставить	Имя	Поставщик	Версия	Описание:
19 5: Нагод У Удалить 19 6: Нагод Рефакторинг 19 7: Нагод Рефакторинг	► TRegul ► TRegul ► TRegul		•	
9 : Нагі — Свойства 9 9 : Нагі — Добавление объекта	Hart Master	"Prosoft-Systems" Ltd.	1.6.5.1	Внешнее устройство (датчи
— 🗊 11 : Hart 📑 Добавить папку				*
12 : Hart Добавить устройство				
13 : Hart Откл. устройство				
15 : Hart				\ .
16: Нат! Редактировать объект В ST_xx_x2x (\$ Редактировать объект в	У Группировать по категориям] Отображать все версии (для эксперто	в) 🔲 Показать устаревшие в
	 Имя: Hart Outer Slave Прокводитель: "Prosoft-Sy Группы: Hart Master Версия: 1.6.5.1 Номер модели: - Описание: Внешнее устройс устройство) с поддержкой Н 	ystems" Ltd. тво (датчик или испольнит art	ельное	▲ E
	Добавить выбранное устройство Hart_Master ① (Можно выбрать другой тарге	о как последнего потоми т-узел, пока окно открыто	.)	
		Добавить устрой	тво	Закрыт

Рисунок 3 – Добавление устройств Hart Outer Slave

Двойным щелчком по названию устройства **Hart Master** откройте вкладку параметров (Рисунок 4).

Hart_Master	8
Настройка Hart Master	Настройки Hart Master
Hart Master МЭК-объектов	Отладочный режим 📃 Вторичный мастер 📃
Состояние	Номер канала в модуле
Информация	
<	-

Рисунок 4 – Параметры устройства Hart Master

Для настройки доступны следующие параметры:

- Отладочный режим установка флажка в этом поле включает режим добавления в журнал контроллера отладочных сообщений HART-мастера;
- Вторичный мастер установка флажка в этом поле задает тип мастера как Вторичное ведущее устройство (Secondary HART Master). По умолчанию устройство работает как Первичное ведущее устройство (Primary HART Master);
- **Номер канала в модуле** номер канала в модуле, к которому подключен и по которому будет опрашиваться устройство Hart Outer Slave.

Двойным щелчком по названию устройства **Hart Outer Slave** откройте вкладку параметров (Рисунок 5).

/1	Hart_Outer_Slave X			•
На	стройка Hart OuterSlave	Общие параметры устройс	ства	_
На	rt команды	Отладочный режим		
Re of	gul Hart Outer Slave МЭК- ьектов	адрес слеива Таймаут слейва	1000	
Co	стояние	Поведение в режиме СТОП	Нет активности 👻	
Ин	формация	Длина преамбулы	8	
		Количество попыток	4	
		Burst режим		
•				•

Рисунок 5 – Параметры устройства Hart Outer Slave

Во вкладке Hactройка Hart Outer Slave доступны следующие параметры:

- Отладочный режим установка флажка в этом поле включает отладочный режим с трассировкой в журнал контроллера;
- Адрес слейва адрес конечного устройства (0-15). Используется в универсальной команде идентификации (Universal Command 0) при начальном опросе устройств на шине;
- Таймаут слейва максимальный таймаут ожидания ответа от HART-устройства, мс;
- Поведение в режиме СТОП указывает, что делать, если переключатель RUN/STOP модуля центрального процессора переведен в положение *STOP*. Возможные значения:
 - Нет активности прекращение опроса,
 - о *Нормальная работа* продолжение работы в обычном режиме,
- Длина преамбулы посылка группы байт 0xFF перед каждой командой, необходимая для синхронизации приемников на шине. Для команды Universal Command 0 всегда используется 20 символов преамбулы;
- Количество попыток число попыток выполнения каждой команды;
- Burst режим установка флажка в поле указывает на присутствие устройства на канале _ в Burst режиме. При попытке добавить еще одно устройство в Burst режиме, на вкладке Настройка Hart Outer Slave высветится знак 🔍. Наведя курсор на знак, появится предупреждающая информация: «Burst режим уже установлен для другого устройства».

ДОБАВЛЕНИЕ ПОЛЬЗОВАТЕЛЬСКИХ КОМАНД, ПЕРЕДАВАЕМЫХ ПО НАRT

Добавление контейнеров HartDevice

Все пользовательские команды, используемые при обмене данными по HART-протоколу, хранятся в специальных объектах **HartDevice**, являющихся контейнерами для команд. Это позволяет группировать команды для конкретных датчиков и других опрашиваемых устройств. Каждый контейнер может содержать множество различных команд. Но к каждому slave-устройству может быть привязан только один контейнер **HartDevice**.

Для добавления контейнера необходимо выполнить следующие действия:

в окне дерева устройств поставьте курсор на объект Application, правой кнопкой мыши вызовите контекстное меню, выберите Добавление объекта ⇒ Устройство HART... (Рисунок б);



Рисунок 6 – Добавление объекта в конфигурацию контроллера

откроется окно Добавить Устройство НАВТ. В поле Имя укажите имя контейнера.
 Например, это может быть название датчика (опрашиваемого устройства) (Рисунок 7).
 Нажмите кнопку Добавить.

Добавить Устройство HART	×
Устройство HART	
Имя: <u>HartDevice_1</u>	
	Добавить Отмена

Рисунок 7 – Добавление контейнера



ИНФОРМАЦИЯ

В дальнейшем при необходимости контейнер можно переименовать (в дереве устройств)

Объект типа HartDevice добавляется в дерево устройств. Автоматически открывается **Редактор hart-устройства** (Рисунок 8).



Рисунок 8 – Редактор hart устройства

Нажмите кнопку *Найти все hart устройства*. В поле **Выберите hart устройство:** отобразится список всех контейнеров типа HartDevice, имеющихся в текущем проекте и во всех подключенных библиотеках (Рисунок 9).

🦻 Редактор hart устройства 🗙				•
Импорт команд из другого hart устройства	ı ———			
Найти все hart устройства				
Выберите hart устройство:	Выберите ко	манды:		📃 Выбрать все
Sensor_2_Hart	Выбрать	Имя	Код	*
DefaultHartDeviceRev7	erre 🗖	Cmn_114	114	
		Cmn_113	113	=
		Cmn_109	109	
		Cmn_108	108	
Импортировать выбранные команды		Cmn_107	107	
		Cmn_106	106	
	🔳	Cmn_105	105	
		Cmn_83	83	
		Cmn_82	82	
		Cmn_81	81	
		Cmn_80	80	
		Cmn_79	79	
		Cmn_76	76	
		Cmn_75	75	
		Cmn_74	74	
		Cmn_73	73	
	····· 🔳	Cmn_72	72	•
(•

Рисунок 9 – Редактор hart устройства

Объект с именем **DefaultHartDeviceRev6/7** – это контейнер с набором универсальных и общих команд, применяемых в спецификации HART Revision 6.0 и HART Revision 7.0 соответственно (для драйвера, реализованного на уровне операционной системы). Префикс **uni** означает, что это команда из спецификации универсальных команд HART (Universal Command Specification), **Cmn** – команда из спецификации общепринятых команд (Common Practice Command Specification). Контейнер является частью библиотеки PsIoDrvHartMaster, которая, в свою очередь, автоматически устанавливается вместе с пакетом, содержащим настройки для обмена по HART-протоколу. Таким образом, пользователю по умолчанию доступен контейнер, из которого можно импортировать нужные команды, и не создавать их вручную.

Использование предустановленного набора команд

Для использования предустановленного набора команд в **Редакторе hart устройства** в поле **Выберите hart устройство:** выберите, например, **DefaultHartDeviceRev6**. В правой части окна в блоке **Выберите команды:** отобразится список всех команд, имеющихся в данном контейнере. Установите флажки для нужных команд и нажмите кнопку *Импортировать выбранные команды*. Команды будут скопированы в пользовательский контейнер (Рисунок 10).

Файл Правка Вид Проект Компиляция Онлайн	Отладка Инструменты Окно Спра	вка			•
🌂 🎦 🗃 📑 📑 🕇 🖻 🐇 🖺 📾 🗙 🔍 🕃 👼 👧	📕 利 🧏 🔞 🔚 🛄 - 🛃 🔛 Aj	pplication [REG	GUL_R500_71	: Plc Logic]	- 🧔 🕸 🕞 🗉 💐
Устройства 👻 🔻 🛪	Редактор hart устройства 🗙				
Hart_project					
🖹 🧔 📅 REGUL_R500_71 (REGUL R500 71)					
Plc Logic	Найти все hart устройства				
Application	Bufanura hart urnaŭrna	Pu fanura va			Pufaati aca
Sensor_1_Hart	высерите нат устроиство:	высерите ко	манды:		выорать все
📑 Cmn_33	Sensor_2_Hart	Выбрать	Имя	Код	*
	DefaultHartDeviceRev7		Cmn_43	43	
📑 Cmn_35			Cmn_42	42	
Sensor_2_Hart			Cmn_41	41	
🔚 Менеджер библиотек			Cmn_40	40	
PLC_PRG (PRG)	Импортировать выбранные команды		Cmn_38	38	
🗈 🔛 Конфигурация задач			Cmn_37	37	
🕮 📲 Regul_Bus (Regul Bus)			Cmn_36	36	
		···· 🔽	Cmn_35	35	
		····· 🔽	Cmn_34	34	
		···· 🔽	Cmn_33	33	
			Uni_22	22	-
			Uni_21	21	=
			Uni_20	20	
			Uni_19	19	
			Uni_18	18	
			Uni_17	17	
		···· 🔳	Uni_16	16	*
🗋 РОU 🎇 Устройства	٠				1
Последняя	компиляция: 🧿 0 🕐 0 🛛 Предкомпил. 🗸	CC CC	Пользоват	тель проекта:	(никто) 🔮 🐼

Рисунок 10 – Импорт команд

Для просмотра параметров команды щелкните дважды левой кнопкой мыши по названию команды в дереве устройств. Откроется Редактор hart команды (Рисунок 11).

🗧 Редактор hart команды 🗙 🗸 🗸 🗸 🗸 🗸										
Параметры команды										
Имя: Стп_33	Iмя: Cmn_33 Результат:									
НАКТ версия: 6 🗸	НАRТ версия: Б Требуемый ф									
Код: 33 🚔				Гребуемый фор	мат структуры приема:	[1,1,4,1,1,4	,1,1,4,1,1,	4,0]		
Описание: Read Device Variable	es									
Формат структуры передачи д	цанных				Формат структуры п	риёма дані	ных			
Добавить новое поле					Добавить новое поле	1				
Имя	Тип	Размер	Описание		Имя		Тип	Размер	Описание	
Slot_0_Device_Variable_Code	BYTE	1	Slot 0 Device	e Variable Code	Slot_0_Device_Varia	ble_Code	BYTE	1	Slot 0: Device Variable Code	
Slot_1_Device_Variable_Code	BYTE	1	Slot 1 Device	e Variable Code	Slot_0_Units_Code		BYTE	1	Slot 0: Units Code	
Slot_2_Device_Variable_Code	BYTE	1	Slot 2 Device	e Variable Code	Slot_0_Device_Varia	ble_Value	REAL	4	Slot 0 Device Variable Value	
Slot_3_Device_Variable_Code	BYTE	1	Slot 3 Device	e Variable Code	Slot_1_Device_Varia	ble_Code	BYTE	1	Slot_1_Device_Variable_Code	
					Slot_1_Units_Code		BYTE	1	Slot_1_Units_Code	
					Slot_1_Device_Varia	ble_Value	REAL	4	Slot_1_Device_Variable_Value	
					Slot_2_Device_Varia	ble_Code	BYTE	1	Slot_2_Device_Variable_Code	
					Slot_2_Units_Code		BYTE	1	Slot_2_Units_Code	
					Slot_2_Device_Varia	ble_Value	REAL	4	Slot_2_Device_Variable_Value	
					Slot_3_Device_Varia	ble_Code	BYTE	1	Slot_3_Device_Variable_Code	
							BYTE	1	Slot_3_Units_Code	
					Slot_3_Device_Varia	ble_Value	REAL	4	Slot_3_Device_Variable_Value	
•					•		III		•	

Рисунок 11 – Редактор hart команды

Дизайн собственных команд

Редактор hart команды

У пользователя может возникнуть необходимость создания контейнера с новыми командами, отсутствующими в подключенных библиотеках, например, при работе с версиями HART, отличными от ревизии 6.0/7.0. В программе предусмотрена возможность редактирования существующих команд и создания собственных новых команд. Также можно разработать и предоставить новую библиотеку с контейнером, содержащим требуемые команды.

Для внесения изменений в существующую команду (импортированную или созданную вручную) щелкните дважды левой кнопкой мыши по названию команды. Откроется окно **Редактор hart команды**, где можно изменить код и описание команды, а также добавить/изменить поля в форматах структур приема/передачи данных (Рисунок 12).

При изменении формата структур для определенного диапазона кодов команд (это команды интегрированного в библиотеку HART-устройства DefaultHartDeviceRev6/7) предусмотрена автоматическая проверка формата структур команды по коду. Если формат структур не соответствует спецификации HART для данного кода команды, то в поле **Результат:** появляется сообщение об ошибке (Рисунок 12). При запуске проекта такая команда не будет выполняться.

/ 📑 Редактор hart команды 🗙	/🔄 Редактор hart команды 🗙 🗸 🗸 🗸 🗸 🗸 🗸 🗸 🗸 🗸 🗸									
Параметры команды			іроверка форм	ата структур					-	
Имя: Стп_34		Pe	езультат:		Ошибка в фо	рмате стр	уктуры перед	ачи		
НАПТ версия: 6	НАRТ версия: 6 ▼			омат структуры передачи:	ат структуры передачи: [4,0]					
Код: 34 🚔		Tp	ребуемый фор	омат структуры приёма:	[4,0]					
Описание: Write Primary Variable Da	amping Value								Ξ	
Формат структуры передачи данных Добавить новое поле Добавить новое поле										
Имя	Тип	Размер	Описані	Имя		Тип	Размер	Описание		
Primary_Variable_Damping_Value	REAL	4	Primary V	···· Primary_Variable_Dar	nping_Value1	REAL	4	Primary Variable Damping		
var_name_2	ARRAY [07] OF LREAL	64								
									-	

Рисунок 12 – Редактор hart команды

Для создания новой команды поместите курсор на название контейнера. Правой кнопкой мыши вызовите контекстное меню и выберите *Добавить объект* ⇒ *HartCommand*... Откроется окно **Добавить HartCommand**, где в поле **Имя:** задайте имя команды (Рисунок 13).

Добавить	HartCommand	23
₿.	HartCommand	
Имя:	HartCommand	
	Добавить Отм	ена

Рисунок 13 – Добавление НАRТ-команды



ИНФОРМАЦИЯ

В дальнейшем при необходимости команду можно переименовать (в дереве устройств)

Нажмите кнопку *Добавить*. Откроется окно **Редактор hart команды** с пустыми полями (Рисунок 14).

/ 📑 Редактор hart команды 🗙	
Параметры команды	Проверка формата структур
Имя: HartCommand	Результат: Ошибка в формате структуры приема
НАRТ версия: 6	Требуемый формат структуры передачи: [0]
Код: 0 👘	Требуемый формат структуры приёма: [1,1,1,1,1,1,1,1,1,1,1,2,1,0]
Описание:	
Формат структуры передачи данных	Формат структуры приёма данных
Добавить новое поле	Добавить новое поле
Имя Тип Размер Описание	Имя Тип Размер Описание

Рисунок 14 – Редактор hart команды при создании новой команды

В поле **HART версия:** выберите значение 6 или 7 из раскрывающего списка, согласно версии файлов описания и библиотек применяемой спецификации - HART Revision 6.0 или HART Revision 7.0 соответственно.

В поле **Код:** с помощью стрелок или вручную введите код команды – номер команды согласно спецификации HART, соответствующий ее функционалу. Произойдет автоматическая проверка команды по коду. Если команда с таким кодом есть в библиотеке, то в блоке **Проверка формата структур** появляется информация о требуемых форматах структур приема/передачи. Сообщение об ошибке присутствует на этом этапе, т.к. форматы пока не определены.

Если команда не описана в подключенных библиотеках, то в поле **Результат:** появится сообщение: «*Проверка структуры отключена*» (Рисунок 15). Для такой команды нет рекомендаций по форматам структур приема/передачи данных. Такая команда будет добавлена на исполнение.

Редактор hart	Редактор hart команды											
Параметры ко	Параметры команды							Проверка формата структур				
Имя:	IR: HartCommand2						Результат:				а структуры отключена	
НАRТ версия: 6 🗸						Требуемый формат структуры передачи: [1,1,0]						
Код:	рд: <u>168</u>					Требуемый формат структуры приёма: [1,1,0]						
Описание:	Описание: Изменение адреса устройства											
Формат стру	уктуры г	тередачи	данных			Формат структуры приёма данных						
Добавить но	овое поле	e				Добавить новое поле						
Имя		Тип	Размер	Описание			Имя	Тип	Размер	Описание		
Polling_A	ddress	USINT	1	Polling Address of D	evice							
Loop		BYTE	1	Loop Current Made	Kon							
					Rom	npo						
					DCIA	БИТ	D					
					Реда	кти	ровать					
L					Удал	ит	ь					

Рисунок 15 – Формирование структур приема/передачи данных

Нажмите кнопку *Добавить новое поле*. Откроется редактор декларации структуры (Рисунок 16).

Редактор де	кларации структуры	8
Имя:	var_name_1	
Тип:	LREAL • Массив? 1	* *
Описание:		
Ok	Отмена	

Рисунок 16 – Редактор декларации структуры

Введите имя команды. Выберите тип поля в соответствии со спецификацией HART или вашими требованиями.

Необязательное поле Описание заполняется по желанию и несет описательную функцию.

Нажмите кнопку *OK*. В Редакторе hart команды появится созданное поле. В дальнейшем эти поля можно редактировать, удалять, копировать с помощью контекстного меню, вызываемого правой кнопкой мыши.

После того, как все поля добавлены/отредактированы, закройте **Редактор hart команды**. Новая команда с заданными параметрами будет добавлена в контейнер и будет отображаться в дереве устройств.

Библиотека PsIoDrvHartMaster

В библиотеке **PsIoDrvHartMaster** для каждой команды, указанной в списке пользовательских команд, определены структуры, где поля соответствуют блоку данных как в запросе, так и в ответе. Дополнительно каждая команда содержит четыре поля (Рисунок 17):

- CmdRespCode;
- CmdDevStatus;
- CmdStatus;
- CmdTrigger.



Рисунок 17 – Дополнительные свойства команды

CmdRespCode и **CmdDevStatus** – после получения ответа от HART-устройства в эти поля копируются два байта Response Code и Field Device Status, которые, согласно спецификации, содержатся в ответе на любую HART-команду; пользователь может самостоятельно проанализировать их содержимое с точки зрения получения расширенной информации о состоянии опрашиваемого устройства.

CmdStatus представляет собой перечисление текущего состояния команды, которое используется в МЭК-приложении для отслеживания цикла обработки команды:

- после добавления команды в очередь она получает статус *Idle* (находится в ожидании начала выполнения по таймеру или по триггеру);
- при переходе в фазу выполнения статус меняется на *InProcess*;
- при успешном получении ответа от устройства и его обработке без ошибок статус устанавливается в значение Ok; при этом в поле данных команды rdata содержится блок данных ответа от устройства и возможен доступ к его отдельным полям согласно структуре конкретной команды;
- в случае ошибки выполнения команды статус может принимать следующие значения:
 - *Timeout* от внешнего HART-устройства не был получен полный ответ на команду в 0 течение периода Slave Timeout, заданного для текущего Hart Outer Slave;
 - о ErrorInResponse ошибка обработки полученного ответа (неверная длина или неверная структура ответа, неверный адрес, команда не реализована в устройстве, устройство занято, наличие ошибки в байте Response Code);
 - о UserReqArg ошибка параметров отдельной динамически формируемой команды, реализованной в функциональном блоке HartUserRequest библиотеки;
 - *InternalError* ошибка, связанная с обработкой команды внутри библиотеки;
- после выполнения команды и получения ответа ее статус не сбрасывается в *Idle*, а сохраняет свое значение до следующего начала цикла обработки и перехода в состояние InProcess;

Полное описание кодов перечисления доступно в МЭК-библиотеке PsIoDrvHartMaster (Таблица 1).

Код	Описание
Idle = 1	В ожидании
InProcess = 2	Идет выполнение цикла "запрос-ответ"
Ok = 3	Данные достоверны, команда успешно выполнена
Timeout = 4	Ошибка - нет ответа
ErrorInResponse = 5	Ошибка - ошибка в ответе
UserReqArg = 6	Ошибка - некорректный аргумент в пользовательском запросе
InternalError = 7	Ошибка - внутренняя ошибка обработки команды

Таблица 1 – Коды перечисления команды Си	mdStatus
--	----------

CmdTrigger – управляющий флаг для команд по требованию (Trigger); в библиотеке фиксируются все переходы триггера из состояния False (0) в состояние True (1) и команда будет добавлена в очередь выполнения заданное число раз.

ОБЪЯВЛЕНИЕ ПЕРЕМЕННЫХ И ПРИВЯЗКА ПЕРЕМЕННЫХ К КОМАНДАМ

Объявление переменных

Для того, чтобы HART-команды выполнялись, требуется добавить в программном коде переменные, представляющие из себя структуры определенных типов, описанных в библиотеке PsIoDrvHartMaster (PsIoDrvHartMaster_OS), и соответствующие добавленным в пункте выше командам для устройства Hart Outer Slave. Для всех добавленных НАRT-команд автоматически генерируются новые типы структур с именами, состоящими последовательно из имени контейнера (HartDevice), затем символа подчеркивания и имени переменной (HartCommand). Например, для контейнера с именем Metran_P_Sensor и командой Read_Primary_Variable будет сгенерирован тип структуры с именем Metran_P_Sensor_Read_Primary_Variable.

Для создания переменных откройте редактор ПЛК-программы (МЭК-приложение). Например, в редакторе ST для программы MAIN создание переменной выглядит следующим образом: *cmd: название контейнера_название команды*. (Рисунок 18).



Рисунок 18 – Пример объявления переменной

Привязка переменных программы к командам

Первым шагом следует привязать контейнер к slave-устройству, далее привязать переменные программы к командам. К одному slave-устройству допустимо привязать только один контейнер с командами.

Двойным щелчком по названию устройства **Hart Outer Slave** откройте редактор устройства. Перейдите на вкладку **Hart Команды** (Рисунок 19).

👘 Hart_Outer_Slave 🗙									•
Настройка Hart OuterSlave	-Hart устрой	ство							
Hart команды	Привязанно	е устрой	ство:						
Regul Hart Outer Slave МЭК- объектов	Быберитен	OBOE nam	сустроиство:	Ŧ					=
Состояние	Команды:								
Информация	Выбрать	Имя	Код команды	Тип вызова	Время цикла (мс)	Переменная для привязки	Описание	Burst режим	
									-

Рисунок 19 – Вкладка «Hart Команды»

В поле **Выберите новое hart устройство:** найдите в раскрывающемся списке название контейнера, выберите его. В поле **Привязанное устройство:** появится название контейнера, а в поле **Команды:** – список команд, которые содержит этот контейнер (Рисунок 20).

👘 Hart_Outer_Slave 🗙										
Настройка Hart OuterSlave	-Hart устро	Hart устройство								
Hart команды	Привязанн	Привязанное устройство: Sensor_1_Hart								
Regul Hart Outer Slave МЭК- объектов	Sensor_1_	Sensor_1_Hart								
Состояние	Команды:									
Информация	Выбрать	Имя	Код команды	Тип вызова	Время цикла (мс)	Переменная для привязки	Описание	Burst режим		
		Uni_0	0	Trigger	-	PLC_PRG.dev1_cmd00	Read Unique Identifier			
	🔽	Uni_1	1	Trigger	-	PLC_PRG.dev1_cmd01	Read Primary Variable	V		
	- · · · · · · · · · · · · · · · · · · ·	Uni_2	2	Trigger	-	PLC_PRG.dev1_cmd02	Read Loop Current And Percent Of Range			
	🔽	Uni_3	3	Trigger	-	PLC_PRG.dev1_cmd03	Read Dynamic Variables And Loop Current			
	····· 🔽	Uni_6	6	Trigger	-	PLC_PRG.dev1_cmd06	Write Polling Address			
	🔽	Uni_7	7	Trigger	-	PLC_PRG.dev1_cmd07	Read Loop Configuration			
	···· V	Uni_8	8	Trigger	-	PLC_PRG.dev1_cmd08	Read Dynamic Variable Classifications			
		11-1-0	0	Tringer		DLC DDC dawt and00	Deniel Device Veriables with Chables	17.8		

Рисунок 20 – Контейнер привязан к slave-устройству

В строке нужной команды установите флажок в поле **Выбрать** и станут доступны для редактирования параметры команды.

В поле Тип вызова команды выбирается, как команда будет добавляться в очередь выполнения, при этом доступны следующие значения для настройки:

- о *заданный период (Timer)* период выполнения команды, мс. Для редактирования будет доступно поле **Время цикла**,
- по требованию (Trigger) для добавления команды в очередь выполнения. Во время выполнения МЭК-приложения каждый раз, когда значение этого поля для переменной, соответствующей добавленной команде, переходит из False в True команда добавляется в очередь.

В поле **Переменная** для привязки можно указать необходимую переменную. Имя переменной можно ввести вручную или через кнопку ..., открывающую окно **Ассистент** ввода. Раскрывая иерархический список, найдите нужную переменную (Рисунок 21).

ссистент ввода				23
Текстовый поиск Категории				
Device variables	Имя	Тип	Адрес	Источник
	E- 1 REGUL_R500_71 Application E- 1 PLC_PRG B- ↑ cmd Config_Globals	Приложение PROGRAM Sensor_1_Hart_Cmn_3 VAR_GLOBAL	4	
	•			•
🔽 Структурированный вид				
Локументация		📝 Вставка с аргументами	Вст	авка с префиксом
cmd: Sensor_1_Hart_Cmn_34(VAR)			*
				-
			ОК	Отмена

Рисунок 21 – Ассистент ввода

Выберите переменную для привязки. Окно Ассистент ввода закроется, а в строке команды в поле Переменная для привязки появится название переменной (Рисунок 22). Для сохранения поместите курсор в любое место окна.

hacipouka hait outerslave	Hart устро	Нагт устройство							
art команды	Привязанн	Привязанное устройство: Sensor_1_Hart							
and that Outer Claus MDK	Выберите новое hart устройство:								
3gul Hart Outer Slave МЭК- Бъектов	Sensor_1	Hart		•					
Состояние Команды:									
нформация	Выбрать	Имя	Код команды	Тип вызова	Время цикла (мс)	Переменная для привязки	Описание	Burst pexa	
		Cmn 33	33				Read Device Variables		
		-					were as we still a solution		
		Cmn_34	34	Timer	1000	PLC_PRG.cmd	Write Primary Variable Damping Value		
		Cmn_34 Cmn_35	34 35	Timer	1000	PLC_PRG.cmd	Write Primary Variable Damping Value Write Primary Variable Range Values		

Рисунок 22 – Переменная для привязки

В поле Burst режим пользователь может установить флажок, указывающий, что данная команда будет использоваться для обработки Burst-пакетов, полученных мастером в сети от соответствующих устройств (см. раздел «Поддержка устройств в Burst режиме»).

К каждой команде можно привязать только переменную соответствующего типа. В случае несовпадения типов появится сообщение об ошибке (Рисунок 23).

Thart_Outer_Slave X									
Настройка Hart OuterSlave	-Hart устрой	ство							
Hart команды	Привязанно Выберите н	Привязанное устройство: Sensor_1_Hart Выберите новое hart устройство:							
Regul Hart Outer Slave МЭК- объектов	Sensor_1_	Sensor_1_Hart							
Состояние	Команды:	Соманды:							
Информация	Выбрать	Имя	Код команды	Тип вызова	Время цикла (мс)	Переменная для привязки	Описание		
	(m. 🗖	Cmn_33	33				Read Device Vari		
	··· 🔽	Cmn_34	34 Actro IDE		-	×	Write Primary Va		
		Cmn_35	35 Astraibe				Write Primary Va		
		HartCommand	122				Изменение адре		
	Ожидаемый тип переменной: Sensor_1_Hart_Cmn_34								
						ОК			

Рисунок 23 – Сообщение об ошибке при несовпадении типов переменных

При изменении привязанного hart-устройства (в поле Выберите новое hart устройство:), будет произведен сброс привязанных переменных без сохранения, подтвердите действие нажатием кнопки *OK* в появившемся окне, иначе *Отмена* (Рисунок 24).



Рисунок 24 – Окно подтверждения сброса привязанных переменных

ОБРАБОТКА ВЫПОЛНЕНИЯ КОМАНДЫ

Обработка выполнения команды по таймеру состоит из следующих шагов:

- проверьте поле статуса выполнения команды. Если статус paвeн CmdStatus.Ok, то команда была выполнена успешно и можно работать с полученными данными;
- ошибка в статусе команды означает сбой при выполнении команды либо наличие ошибки в самом ответе от устройства и требуется дополнительный анализ кода CmdStatus либо анализ полей команды CmdRespCode и CmdDevStatus.

Обработка выполнения команды по триггеру состоит из следующих шагов:

для добавления команды в очередь выполнения по триггеру установите в коде программы для поля CmdTrigger команды значение True (предварительно это поле должно быть выставлено в False):

```
IF Условие запуска команды на выполнение = TRUE THEN
      P Sensor cmd01.CmdTrigger := TRUE;
END IF
```

- отслеживайте успешное выполнение команды по полю статуса:
 - о если статус paben CmdStatus.Ok, то команда в последнем цикле была выполнена успешно и можно работать с полученными данными;
 - о ошибка в статусе команды означает сбой при выполнении команды либо наличие ошибки в самом ответе от устройства и требуется дополнительный анализ кода CmdStatus либо анализ полей команды CmdRespCode и CmdDevStatus.

Обработка полученных значений в поле данных команды:

```
IF P Sensor cmd01.CmdStatus = CmdStatus.Ok THEN
        byteValue1 := P_Sensor_cmd01.rx_data.Primary_Variable_Units;
realValue2 := P_Sensor_cmd01.rx_data.Primary_Variable;
END IF
```

Обработка выполнения Burst команды состоит из следующей последовательности действий:

- проверьте поле статуса выполнения команды. Если статус равен CmdStatus.Ok, то Burst пакет был обработан успешно и можно работать с полученными данными;
- ошибка в статусе команды CmdStatus.ErrorInResponse означает сбой при обработке Burst пакета, либо наличие ошибки в самом ответе от устройства, и требуется дополнительный анализ кода CmdStatus, либо анализ полей команды CmdRespCode и CmdDevStatus.



ИНФОРМАЦИЯ

При выполнении пользовательских команд используется «длинный» (5 байт) уникальный адрес устройства, который формируется в библиотеке при получении ответа на запрос команды идентификации (Universal Command #0). Поэтому, если уникальный адрес еще не сформирован, то перед выполнением пользовательской команды библиотека автоматически добавляет команду идентификации и пытается ее выполнить. Также эта команда будет автоматически выполняться в случае ошибки обмена с устройством перед следующей в очереди пользовательской командой

ПОДДЕРЖКА УСТРОЙСТВ В BURST РЕЖИМЕ

Burst режим позволяет устройству циклически отправлять в сеть оперативные данные без запроса со стороны мастера. Только одно устройство в сети может быть переведено в этот режим. Для настройки Burst режима используются команды, описанные в таблице 2:

Таблица 2.	– Команлы	ппа н	астройки	Rurst	пежима
1 аолица <i>2</i> -	- команды	для н	астроики	Duisi	режима

Команда	Описание
Command 109 Burst Mode Control	Включение и выключение Burst режима
Command 108 Write Burst Mode Command Number	Задает выбор номера команды, ответ на которую устройство будет транслировать в сеть. К таким командам относятся команды 1, 2, 3, 9 и 33. Команда может выполняться как на этапе конфигурации устройства, так и в процессе работы в режиме Burst «на лету»
Command 107 Write Burst Device Variables	Позволяет выбрать номера Device Variables в устройстве, которые будут переданы в ответе на команду 9 и 33
Command 105 Read Burst Mode Configuration	Чтение текущей конфигурации Burst

Для организации получения данных от устройства в Burst режиме, подключенном на один из каналов, необходимо выполнить следующие действия:

- 1. Добавьте HART-устройство на канал (**HART Master**⇒**Hart Outer Slave**). Установите в общих параметрах устройства флажок в поле **Burst режим**.
- 2. Добавьте контейнер HartDevice (Application ⇒Добавление объекта... ⇒Устройство HART...). На вкладке Редактор hart устройства выберите hart устройство и импортируйте в контейнер необходимые для настройки (105,107,108,109) и получения данных (1,2,3,9,33) команды.
- 3. Перейдите на вкладку **Hart команды** объекта **Hart Outer Slave** (п.1). Активируйте созданный в п.2 контейнер команд, установив флажок в поле **Выбрать**. Далее установите флажок в поле **Burst режим** для тех команд, по которым будут приходить оперативные данные от устройства в Burst режиме.



ВНИМАНИЕ!

В контейнер команд **HartDevice** можно добавить только один экземпляр команды каждого конкретного типа. При этом команда может обрабатываться либо в обычном режиме (запрос/ответ, циклически или по триггеру), либо в Burst режиме (получение пакетов от Burst устройства в режиме прослушивания сети, только команды 1, 2, 3, 9, 33), в зависимости от состояния флажка команды из п.3. Тип вызова команд не важен, если выставлен флажок в поле Burst режим

4. Добавьте в код переменные соответствующих типов (PLC_PRG). В экземпляры этих переменных будут копироваться данные из полученных Burst пакетов. В списке команд из п.3 на вкладке **Hart команды** привяжите добавленные переменные к активным командам с помощью ассистента ввода в поле **Переменная для привязки.** (Рисунок 25).

To Hart_Outer_Slave X										
Настройка Hart OuterSlave	Hart устройство									
Hart команды	Привязанное устройство: Sensor_1_Hart									
Regul Hart Outer Slave МЭК- объектов Состояние	Sensor_1_Hart									
Информация	Выбрать	Имя	Код команды	Тип вызова	Время цикла (мс)	Переменная для привязки	Описание	Burst режим		
	· · · · · · · · · · · · · · · · · · ·	Uni_1	1	Timer	1000	PLC_PRG.dev1_cmd01	Read Primary Variable			
	V	Uni_2	2	Timer	1000	PLC_PRG.dev1_cmd02	Read Loop Current And Percent Of Range	\checkmark		
	V	Uni_3	3	Timer	1000	PLC_PRG.dev1_cmd03	Read Dynamic Variables And Loop Current			
		Uni_9	9	Timer	1000	PLC_PRG.dev1_cmd09	Read Device Variables with Status	\checkmark		
	····· 🔽	Cmn_33	33	Timer	1000	PLC_PRG.dev1_cmd33	Read Device Variables	\checkmark		
	···· 🔽	Cmn_105	105	Trigger	-	PLC_PRG.dev1_cmd105	Read Burst Mode Configuration			
	···· 🔽	Cmn_107	107	Trigger	-	PLC_PRG.dev1_cmd107	Write Burst Device Variables			
	···· 🔽	Cmn_108	108	Trigger	-	PLC_PRG.dev1_cmd108	Write Burst Mode Command Number			
	· 📝	Cmn_109	109	Trigger	-	PLC_PRG.dev1_cmd109	Burst Mode Control			

Рисунок 25 - Добавление и привязка команд для Burst режима

5. После загрузки и запуска приложения на контроллере экземпляр объекта Hart Outer Slave переходит в рабочий режим – выполнение обычных команд (запрос/ответ) и прослушивание сети на предмет обнаружения Burst пакетов. Уникальность команд в контейнере HartDevice, привязанном к объекту Hart Outer Slave, позволяет однозначно сопоставить экземпляр переменной МЭК-приложения и данные в полученном Burst пакете. В случае успешной обработки пакета, данные из него, включая байты Response Code и Field Device Status, копируются в соответствующие поля структуры переменной МЭК-приложения, а поле CmdStatus принимает значение Ok (3). В случае ошибки поле CmdStatus устанавливается в значение ErrorInResponse (5).

ФУНКЦИОНАЛЬНЫЙ БЛОК HARTUSERREQUEST И ДИНАМИЧЕСКИ ФОРМИРУЕМАЯ КОМАНДА

Общее описание

В библиотеке имеется возможность работы в МЭК-приложении с динамически формируемой командой HART, реализованной в функциональном блоке (далее – ФБ) *HartUserRequest*. Работа с командой – только по триггеру. Burst команды не поддерживаются.

Менеджер библиотек								8
🗄 Добавить библиотеку 🗙 Удалить библиотеку 🖙 Свойства 🗃 Детали 🔄 Плейсхолдеры 🎁 Репозиторий библиотек 🕕 Легенда иконки 💿 🍃								
Имя					Дополнительное имя	Действующая версия	-	
🕮 🖸 3SLicense = 3SLicense, 3.5. 17.0 (3S - Smart Software Solutions GmbH)					_3S_LICENSE	3.5.17.0		
BreakpointLogging = Breakpoint Logging Functions, 3.5.17.0 (3S - Smart Software Solutions GmbH)					BPLog	3.5.17.0	=	
CDS_MemMan = CoDeSys Memory Manager, 3.5.17.0 (3S - Smart Software Solutions GmbH)					CMM	3.5.17.0		
🗄 🗄 IoStandard = IoStandard, 3.5	5.17.0 (System)					IoStandard	3.5.17.0	
🖳 🖳 PsIoDrvHartMaster_OS = PsIo	DrvHartMaster_OS, 1.7.0.0	(Prosoft-Systems Ltd)				PsIoDrvHartMaster_OS	1.7.0.0	
B: PstoDrvReaulBus = PstoDrvRe	eaulBus. 1.7.0.0 (Prosoft-Svs	tems Ltd)				PsToDrvReaulBus	1.7.0.0	-
PsIoDrvHartMaster_OS, 💌	- Входы/Выходы 🚮 г	рафич. ? Документа	ция					
CmdBase			-					
	FUNCTION_BLOCK Har	tUserRequest						
	Имя	Тип	H .	A. I	н.	Комментарий		
	*≱ xExecute	BOOL				М: Управление работой ФБ		
	✤ refHartOuterSlave	REFERENCE TO Hart				М: Ссылка на опрашиваемое	е устройство	_
🗉 📝 DefaultHartDeviceRev6	🔖 byCommand	BYTE				М: Код команды		
🗉 📝 DefaultHartDeviceRev7	🐐 byRepeatNum	BYTE				М: Число повторных запрос	ов	
HartUserRequest	🐐 sSpecificName	STRING		Ъ	lart	О: Имя запроса		
IoDr∨PsHartMaster	💖 pbyRDataFormat	POINTER TO BYTE				М: Указатель на формат да	нных для приема	
	💖 byRDataFormatLen	BYTE				М: Размер массива формата	данных для приема в байтах	
	🔖 pbyXDataFormat	POINTER TO BYTE				М: Указатель на формат да	нных для передачи	
	🔖 byXDataFormatLen	BYTE				М: Размер массива формата	данных для передачи в байтах	
	🔖 pbyRData	POINTER TO BYTE				М: Буфер данных для прием	ıa	
	🔖 byRDataLen	BYTE				М: Размер буфера данных д	ля приема в байтах	
	🔖 pbyXData	POINTER TO BYTE				М: Буфер данных для перед	ачи	
	🔖 byXDataLen	BYTE				М: Размер буфера данных д	ля передачи в байтах	
	🔷 xDone	BOOL				Запрос выполнен успешно		
	🔷 xError	BOOL				Ошибка выполнения запрос	a	
	🔷 eStatus	CmdStatus				Код ошибки		
	bRespCode	BYTE				Код ответа		
	bDevStatus	BYTE				Статус устройства		
4 III >								

Рисунок 26 – Функциональный блок HartUserRequest

Для корректной обработки произвольной HART-команды в переменную ФБ *HartUserRequest* при ее вызове в МЭК-приложении передаются специфические параметры, включающие форматы структур поля данных для записи (в запросе) и чтения (в ответе). Это позволяет библиотеке корректно отформатировать выходной/входной блок данных запроса/ответа и сопоставить разнотипные параметры, содержащиеся в этом блоке, соответствующим полям переменной команды, объявленной в МЭК-приложении.

Ниже приведен пример выполнения универсальной команды запроса информации об устройстве (команда с кодом 15). При этом доступны два варианта задания структуры данных.

Объявление структуры данных команды непосредственно в коде

Для объявления структуры данных команды непосредственно в коде необходимо наследовать структуру 15-й команды от базовой структуры CmdBase, объявленной в библиотеке (Рисунок 27).

```
//UNI 15 - Read Device Information;
TYPE cmd15 EXTENDS PsIoDrvHartMaster.CmdBase :
STRUCT
           data : cmd15 rdata; //UNI 15 - Read Device Information;
END STRUCT
END TYPE
```

Добавить библиотеку Х Удалить библиотеку				
Имя Дополнит				
Hull 20 icense = 20 icense - 2.5.17.0 (25 - Smart Software Solutions CmbH) 25. LICENS	ельное ик 🔺			
Salcense – S	E Master OS			
енистания еАttr Имя Тип Н. А., Н. Комментарий				
TaskInfo 🛛 🖗 RespCode BYTE Communication Status/Response Cod	e;			
UserReqRet / DevStatus BYTE Field Device Status;				
DefaultHartDev Status CmdStatus Command execute status;	Command execute status;			
Trigger BOOL Start command trigger;				

Рисунок 27 – Базовая структура CmdBase

С полем данных типа структуры cmd15 rdata, соответствующей спецификации HART.

```
{attribute 'pack mode' := '1'}
TYPE cmd15 rdata :
                                                                                        //UNI 15 - Read Device Information;
STRUCT
             FPV_AlarmSelCode : BYTE;PV_TransFuncCode : BYTE;PV_UL_UnitCode : BYTE;PV_UL_UnitCode : BYTE;PV_UpperRangeValue: REAL;PV_UpperRangeValue: REAL;PV_LowerRangeValue: REAL;PV_DampingValue : REAL;PV_DampingValue : REAL;PV_DampingValue : REAL;PV_DampingValue : REAL;PV_V Damping Value;PrivLblDistrCode : BYTE;PV_AnalogChFlags : BYTE;PV_PV Analog Channel Flags;
END STRUCT
END TYPE
```

Далее, в области объявлений программного блока (POU) обработки HART-команды следует объявить набор рабочих переменных.

//экземпляр FB пользовательского запроса; UserRequest : PsIoDrvHartMaster.HartUserRequest;

```
//код команды;
     bUR Command : BYTE := 0;
     //число попыток выполнения команды;
     bUR RepeatNum : BYTE := 4;
     //выходной параметр ФБ - флаг успешного выполнения команды;
     xUR_Done : BOOL := FALSE;
     //выходной параметр ФБ - флаг ошибки выполнения команды;
     xUR Error : BOOL := FALSE;
     //размер структуры данных ответа;
     bUR CommandRxDataLen: BYTE;
      //размер структуры данных запроса;
     bUR CommandTxDataLen: BYTE;
     //флаг управления запуска/остановки команды;
     xUR Execute Uni 15 : BOOL := FALSE;
     //структура команды;
     cmdUR Command 15 : cmd15;
      //формат блока данных ответа от устройства, содержащий размер полей данных в
байтах;
     arrbUR CommandRxFrm 15: ARRAY[0..9] OF BYTE := [1, 1, 1, 4, 4, 4, 1, 1, 1,
0];
      //количество полей блока данных ответа, включая завершающий '0';
     bUR CommandRxFrmtLen 15: BYTE := 10;
      //формат блока данных запроса к устройству, содержащий размер полей данных в
байтах;
     arrbUR CommandTxFrmt 15: POINTER TO BYTE; //для этой команды не используется;
     //количество полей блока данных запроса, включая завершающий '0';
     bUR CommandRTxFrmtLen 15: BYTE;
                                                           //для этой команды не
```

```
используется;
```

В теле блока обработка команды сводится к следующим строкам:

```
// Выполнение пользовательского запроса
IF xUR Execute_Uni_15 = TRUE THEN
     bUR Command := 15;
     bUR CommandRxDataLen := UINT TO BYTE(SIZEOF(cmdUR Command 15.data));
     bUR CommandTxDataLen := 0;
     UserRequest ( xExecute := xUR Execute Uni 15,
                 refHartOuterSlave := Hart Outer Slave,
                 byCommand := bUR Command,
                 byRepeatNum := bUR RepeatNum,
                 sSpecificName := 'Test',
                 pbyRData := ADR(cmdUR Command 15.data),
                 byRDataLen := bUR CommandRxDataLen,
                 pbyRDataFormat := ADR(arrbUR CommandRxFrm 15),
                 byRDataFormatLen := bUR CommandRxFrmtLen 15,
                 bRespCode => cmdUR Command 15.RespCode,
                 bDevStatus => cmdUR Command 15.DevStatus,
                 eStatus => cmdUR Command 15.Status,
                 xDone => xUR Done,
                 xError => xUR Error
     );
     IF xUR Done OR xUR Error THEN
           xUR Execute Uni 15 := FALSE;
           UserRequest ( xExecute := xUR Execute Uni 15);
     END IF
END IF
```

В данном случае Hart Outer Slave – это имя экземпляра функционального блока, соответствующего устройству Hart Outer Slave в дереве устройств.

Использование команды из контейнера команд HART-устройства

Для использования команды из контейнера команд HART-устройства необходимо создать в дереве устройств фиктивный контейнер и добавить/импортировать в него 15-ю команду, для которой автоматически создается тип структуры HartUserRequest Uni 15 (Рисунок 28).

Устройства 👻 🕂 🗙	📑 Редактор hart команды 🗙			Ŧ	
Hart project	Параметры команды Има: Uni_15 НАRТ версия: 6 Код: 15 (С) Описание: Read Device Information Формат структуры передачи данных Добавить новое поле	Проверка формата структур Результат: Ok Требуемый формат структуры передачи: [0] Требуемый формат структуры приёма: [1,1,1,4,4,4,1,1,1,0] Формат структуры приёма данных			
< III • • • • • • • • • • • • • • • • •	Имя Тип Размер Описание	Имя PV_Alarm_Selection_Code PV_Transfer_Function_Code PV_Upper_and_Lower_Range_Values_Units_Code PV_Upper_Range_Value PV_Lower_Range_Value PV_Damping_Value Write_Protect_Code Private_Label_Distributor_Code PV_Analog_Channel_Flags	TunPaBYTE1BYTE1BYTE1REAL4REAL4BYTE1BYTE1BYTE1	B3	
	Последняя компиляция: 😋 0 🕐 0 🛛 Предкомп	пил. 🗸 🤹 Пользователь проекта: (никто)	6 🚯		

Рисунок 28 – Структура HartUserRequest Uni 15

Далее, в области объявлений программного блока (POU) обработки HART-команды следует объявить набор рабочих переменных.

```
//экземпляр FB пользовательского запроса;
     UserRequest
                           : PsIoDrvHartMaster.HartUserRequest;
     //код команды;
     bUR Command : BYTE := 0;
     //число попыток выполнения команды;
     bUR RepeatNum : BYTE := 4;
     //выходной параметр ФБ - флаг успешного выполнения команды;
     xUR_Done : BOOL := FALSE;
     //выходной параметр ФБ - флаг ошибки выполнения команды;
                        : BOOL := FALSE;
     xUR Error
     //размер структуры данных ответа;
     bUR CommandRxDataLen: BYTE;
     //размер структуры данных запроса;
     bUR CommandTxDataLen: BYTE;
     //структура команды;
     cmd 15 : HartUserRequest_Uni_15;
     //формат блока данных ответа от устройства, содержащий размер полей данных в
байтах;
     arrbUR CommandRxFrm 15: ARRAY[0..9] OF BYTE := [1, 1, 1, 4, 4, 4, 1, 1, 1,
0];
     //количество полей блока данных ответа, включая завершающий '0';
     bUR CommandRxFrmtLen 15: BYTE := 10;
     //формат блока данных запроса к устройству, содержащий размер полей данных в
байтах;
     arrbUR CommandTxFrmt 15: POINTER TO BYTE; //для этой команды не используется;
```

//количество полей блока данных запроса, включая завершающий '0'; bUR CommandRTxFrmtLen 15: BYTE; //для этой команды не используется;

В теле блока обработка команды сводится к следующим строкам:

```
// Выполнение пользовательского запроса
IF cmd 15.CmdTrigger = TRUE THEN
      bUR Command := 15;
      bUR CommandRxDataLen := UINT TO BYTE(SIZEOF(cmd 15.rx data));
      bUR CommandTxDataLen := 0;
                      xExecute := cmd_15.CmdTrigger,
      UserRequest(
                   refHartOuterSlave := Hart Outer Slave,
                   byCommand := bUR Command,
                   byRepeatNum := bUR_RepeatNum,
                   sSpecificName := 'Test',
                   pbyRData := ADR(cmd 15.rx data),
                   byRDataLen := bUR CommandRxDataLen,
                   pbyRDataFormat := ADR(arrbUR_CommandRxFrm_15),
byRDataFormatLen := bUR_CommandRxFrmtLen_15,
                   bRespCode => cmd 15.CmdRespCode,
                   bDevStatus => cmd 15.CmdDevStatus,
                   eStatus => cmd 15.CmdStatus,
                   xDone => xUR Done,
                   xError => xUR Error
      );
      IF xUR Done OR xUR Error THEN
            cmd_15.CmdTrigger := FALSE;
            UserRequest( xExecute := cmd_15.CmdTrigger);
      END IF
END IF
```

Пример выполнения команды 34 – Write Primary Variable Damping Value

Для объявления структуры данных команды непосредственно в коде необходимо наследовать структуру 34-й команды от базовой структуры CmdBase, объявленной в библиотеке.

```
//COMMON PRACTICE 34 - Write Primary Variable Damping Value;
TYPE cmd34 EXTENDS PsIoDrvHartMaster.CmdBase:
STRUCT
                            //COMMON PRACTICE 34 - OUT - Write Primary Variable
     xdata : cmd34 data;
Damping Value;
                            //COMMON PRACTICE 34 - IN - Write Primary Variable
     rdata : cmd34 data;
Damping Value;
END STRUCT
END TYPE
  Гле:
{attribute 'pack mode' := '1'}
TYPE cmd34 data : //COMMON PRACTICE 34 - Write Primary Variable Damping Value;
STRUCT
      PV DampungValue : REAL;//Primary Variable Damping Value (units of seconds);
END STRUCT
```

END TYPE

Далее, в области объявлений программного блока (POU) обработки HART-команды следует объявить набор рабочих переменных.

```
//экземпляр FB пользовательского запроса;
     UserRequest
                                  : PsIoDrvHartMaster.HartUserRequest;
     //код команды;
     bUR Command
                                 : BYTE := 0;
     //число попыток выполнения команды;
     bUR RepeatNum : BYTE := 4;
     //выходной параметр ФБ - флаг успешного выполнения команды;
                              : BOOL := FALSE;
     xUR Done
     //выходной параметр ФБ - флаг ошибки выполнения команды;
                                : BOOL := FALSE;
     xUR Error
     //размер структуры данных ответа;
     bUR CommandRxDataLen: BYTE;
     //размер структуры данных запроса;
     bUR CommandTxDataLen: BYTE;
     //флаг управления запуска/остановки команды;
     xUR Execute Cmn 34 : BOOL := FALSE;
     //структура команды;
     cmdUR Command 34 : cmd34;
     //формат блока данных ответа от устройства, содержащий размер полей данных в
байтах;
     arrbUR CommandRxFrmt 34: ARRAY[0..1] OF BYTE := [4, 0];
     //количество полей блока данных ответа, включая завершающий '0';
     bUR CommandRxFrmtLen 34: BYTE := 2;
     //формат блока данных запроса к устройству, содержащий размер полей данных в
байтах;
     arrbUR CommandTxFrmt 34: ARRAY[0..1] OF BYTE := [4, 0];
     //количество полей блока данных запроса, включая завершающий '0';
     bUR CommandTxFrmtLen 34: BYTE := 2;
```

В теле блока обработка команды сводится к следующим строкам:

```
IF xUR Execute Cmn 34 = TRUE THEN
     bUR Command := 34;
     bUR CommandRxDataLen := UINT TO BYTE(SIZEOF(cmdUR Command 34.rdata));
     bUR CommandTxDataLen := UINT TO BYTE(SIZEOF(cmdUR Command 34.xdata));
     cmdUR Command 34.xdata.PV DampungValue := 4.0;
                       xExecute := xUR_Execute Cmn 34,
     UserRequest(
                 refHartOuterSlave := Hart Outer Slave,
                 byCommand := bUR Command,
                 byRepeatNum := bUR RepeatNum,
                 sSpecificName := 'Test',
                 pbyRData := ADR(cmdUR Command 34.rdata),
                 byRDataLen := bUR CommandRxDataLen,
                 pbyRDataFormat := ADR (arrbUR CommandRxFrmt 34),
                 byRDataFormatLen := bUR CommandRxFrmtLen 34,
                 byXDataLen := bUR CommandTxDataLen,
                 pbyXData := ADR (cmdUR Command 34.xdata),
                 pbyXDataFormat := ADR (arrbUR CommandTxFrmt 34),
                 byXDataFormatLen := bUR CommandTxFrmtLen 34,
                 bRespCode => cmdUR Command 34.RespCode,
                 bDevStatus => cmdUR Command 34.DevStatus,
                 eStatus => cmdUR Command 34.Status,
                 xDone => xUR Done,
                 xError => xUR Error
     ) :
     IF xUR Done OR xUR Error THEN
           xUR Execute Cmn 34 := FALSE;
           UserRequest ( xExecute := xUR Execute Cmn 34);
     END IF
END IF
```