Преимущества VS EtherCAT Master 2

Новое поколение ядра VS EtherCAT Master (Master) предоставляет исключительные возможности автоматизации управления сетью EtherCAT. Master 1 имеет множество ограничений из-за своей негибкой и монолитной архитектуры. Это может привести к трудностям в его использовании для некоторых целевых систем, например:

- для целевых систем без установленной ОС
- для целевых систем с ограничением количества потоков в одном процессе
- для целевых систем с нехваткой системных ресурсов

В новом ядре Master мы пересмотрели его архитектуру и приложили максимальные усилия, чтобы избежать всех ограничений предыдущих версий, а также улучшили и расширили функциональность Master. Master 2 сохраняет совместимость с уровнем API Master 1. Ниже приведены описания возможностей Master 2.

Асинхронное управление

В Master 2 реализована новая модель управления, называемая «асинхронное управление», которая позволяет параллельно управлять выполнением операций, осуществляемых Master и пользователем, а также других функций.

Параллельное выполнение операций. Асинхронная модель позволяет не прерывать поток выполнения пользовательских операций, пока Master выполняет вызов API. Пользователь может продолжать работу и проверять результат API вызовов, когда это необходимо.

Отложенные операции. Асинхронный режим не предполагает немедленного результата, Master выполняет операцию по мере готовности; возвращенный код и данные сохраняются в дескрипторе операции. В зависимости от ситуации пользователь может либо дождаться результата операции, либо продолжить работу и проверить завершена ли операция, когда это будет необходимо.

Несколько операций. Асинхронная модель позволяет асинхронно выполнять несколько операций и ждать их завершения. Асинхронное управление требует использования уникального идентификатора (транзакции, сеанса и т. д.), который напрямую связывает ожидающий запрос и результат операции. Master использует дескриптор объекта в качестве такого идентификатора. Все асинхронные операции всегда привязаны к определенному экземпляру объекта. Это позволяет получить текущее состояние запроса и результат операции.

Очередь синхронизации используется для получения уведомлений об обработке асинхронных вызовов; это помогает объединить несколько операций, связанных с разными функциями, в одну очередь ожидания. Есть возможножность управления всем Master в одной очереди синхронизации (запуская «однопоточную» модель).

В Мастере 1.6 при синхронном выполнении операций пользователю приходится ждать завершения вызова API, и при этом нет возможности выполнять какую-либо работу до завершения вызова API. Асинхронная модель позволяет не прерывать поток выполнения работы, пока Master выполняет вызов API. Пользователь может продолжать работу и проверять результат API вызовов, когда это необходимо.

Выполнение простых задач

Master 2 использует задачно-ориентированную модель, при которой происходит разделение операций на простые задачи (Master Simple Tasks (MST)). В этом случае MST является

наименьшим элементом исполнения, т.е. его невозможно остановить. Ниже приведен список функций, основанных на этой модели.

Выполнение простых задач. Затрачивается минимальное время выполнения и быстро происходит реакция на запрос.

Имитируемый многопоточный режим. Разделив операции на MST, их можно выполнять параллельно, что имитирует многопоточный режим Master. Порядок выполнения MST определяется их приоритетом.

Многопоточность. Master 2 позволяет использовать любое количество потоков Master для выполнения MST. В этом случае все MST закреплены за четырьмя исполнителями в зависимости от их назначения: реальное время, фон, синхронизация и система. Затем исполнители назначаются разным потокам с разным приоритетом. По сравнению с предыдущими версиями Master, где можно было использовать только три потока, такой подход приносит следующие преимущества:

Распределение нагрузки на ядра. Многопоточность позволяет оптимально распределить нагрузку на ядра, назначая им разное выполнение простых задач.

Одновременное выполнение задач. MST, назначенные разным потокам, могут выполняться одновременно.

Несколько циклических задач EtherCAT

Помимо Master Simple Tasks, Master 2 поддерживает несколько циклических задач EtherCAT (EtherCAT Cyclic Tasks (ECT)). VS EtherCAT Studio предоставляет возможность генерировать ECT. Эти задачи могут контролироваться Master или внешним приложением. ECT присвоены определенным данным процесса и имеют определенный цикл выполнения.

Используя ECT, пользователь может запускать их с разными циклами и в разное время. Это позволяет контролировать загрузку шины путем назначения операций с разными циклами выполнения ECT.

Модульная конструкция

Модульная конструкция делает ядро Master невероятно гибким и легко настраиваемым. Почти все части, включая основные компоненты спроектированы как отдельные модули. Это позволяет создать ядро Master в соответствии с потребностями пользователя и для достижения оптимального баланса производительности и функциональности. Ниже приведен список преимуществ модульной конструкции.

Разделение функциональности на отдельные модули. Каждый модуль спроектирован как изолированный блок со своими API, переменными PI, событиями, конфигурацией через INI-файл и документацией.

Расширение функциональности. Для каждой новой функциональности следует создавать новый модуль. Оптимизация. Ядро Master также легко оптимизировать путем отключения модулей или их замены.

Управление PI

Master 2 реализует новый подход к доступу и работе с данными в образе процесса (process image (PI)). Доступ к PI выполняется клиентом PI. Это объект, который обеспечивает доступ к PI для чтения/записи, доставляет данные от PI к внутреннему буферу клиента, и который можно подписать на события и изменения PI. Новый интерфейс позволяет нескольким пользователям

получить доступ к PI одновременно. Ниже приведены преимущества управлением PI.

Защищенный доступ к PI. Каждая часть PI может принадлежать только одному клиенту, что ограничивает доступ на запись в PI. Только один клиент может записывать данные в указанную область PI, всем остальным это запрещено.

Несколько клиентов PI. Пользователь может создать любое количество клиентов PI. Каждый из них будет сопоставлен с определенной областью PI, подписан на разные события и будет иметь разные права и изолированный внутренний буфер.

Сопоставление переменных PI со структурой данных пользователя. Пользователь может сопоставить определенные переменные PI со своей структурой и далее работать только со своей структурой без прямого вызова переменных PI.

Атомарные операции. Данные клиента PI обрабатываются одной непрерывной задачей. Это гарантирует согласованность данных.

Несколько буферов доставки. Это режим, когда клиент PI использует несколько буферов при выполнении операции. Например, когда Master уже выполнил операцию, но приложение пользователя все еще занято и не может получить данные операции. В этом случае клиент PI может использовать несколько буферов доставки и хранить данные, полученные на каждой итерации, в разных буферах. Когда приложение пользователя освобождается, оно последовательно извлекает полученные данные из буферов. Таким образом, это гарантирует сохранность всех полученных данных.

Доставка по записи или по событию. Благодаря асинхронному интерфейсу и управлению PI нет необходимости для пользовательского приложения запрашивать данные Master в каждом цикле Master. Теперь пользователь может один раз настроить Master на отправку определенных (обязательных) данных одним из способов:

по доступу на запись - когда происходит доступ на запись к определенной области в PI

по событию - когда происходит определенное событие

Диагностические данные в образе процесса

По сравнению с Master 1, в Master 2 все диагностические данные расположены в образе процесса. Такой подход позволяет получать диагностические данные одновременно с данными процесса. Это не требует дополнительных вызовов для получения диагностики, что упрощает работу Master.

Теперь пользователь может наблюдать, как изменения данных процесса влияют на изменения диагностических данных. Чтобы визуализировать изменения диагностических данных можно получить снимки соответствующих сигналов и просмотреть их в инструменте Run-time Data Logger в VS EtherCAT Studio.

PI Logger

Теперь, используя несколько клиентов PI, можно отслеживать изменения любых переменных PI в режиме выполнения с помощью инструмента Run-time Data Logger VS EtherCAT Studio. Master делает снимки настроенных переменных PI (пользовательских, служб Slave или Master, и т. д.) для необходимого события, а затем сохраняет несколько снимков в буфер и отправляет их в Studio. Пользователь может анализировать изменения переменных, используя инструмент Run-time Data Logger (в Studio), где они представлены в виде диаграммы.

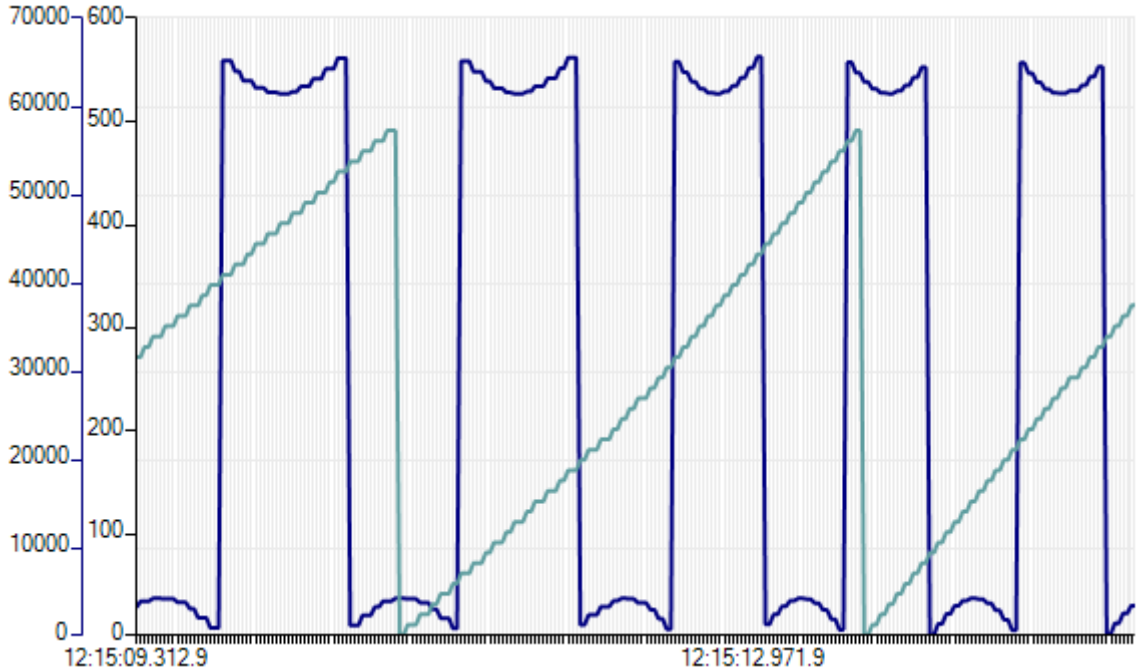


Figure 1. Пример диаграммы переменных в Run-time Data Logger

PI Logger можно использовать в качестве поставщика данных для сторонних инструментов, таких как MATLAB или LabVIEW, или любых других инструментов для дальнейшего расширенного анализа на основе Python.

События

По сравнению с Master 1, где события использовались в функции обработчика событий (опциональной и требующей дополнительного лицензирования), ядро Master 2 основано на модели событий. Все процессы Master активируются/выполняются событиями. Каждый модуль Master содержит события, которые могут использоваться как в Master, так и управляющим приложением.

Событие — это тип сообщения, которое Master генерирует в некоторых точках выполнения, когда необходимо информировать клиента об изменениях в состоянии Master, конфигурации или обработке. События могут уведомлять об ошибках, предупреждениях или просто предоставлять информацию. Все события генерируются в одинаковом формате.

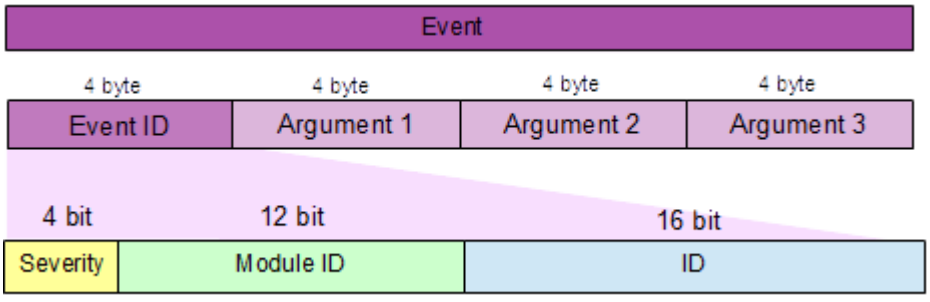


Figure 2. Event format

Где
Severity – серьезность события, допустимые значения: INFO, WARNING, ERROR или TRACE (можно использовать для пользовательских событий).
Module ID – идентификатор модуля Master, который генерирует соответствующее уведомление о событии, например, модуля Master для диагностики топологии шины (состояние шины,

подключение устройств и т. д.) или ведомых устройств (их текущее и запрошенное состояние, неудачные команды инициализации и т. д.) или Master (его состояние, недействительная конфигурация, нарушенные соединения и т. д.).
ID – идентификатор события в модуле Master.
***Argument 1, Argument 2, Argument 3** — параметры, специфичные для события.

События используются Master для синхронизации выполнения задач, уведомления о циклических событиях, событиях состояния или чрезвычайных событиях. Все события Master доступны на уровне пользовательского приложения. Это позволяет приложению реализовать режим событийно-ориентированного выполнения. Большую часть времени приложение находится в спящем режиме и не требует ресурсов процессора. Как только произойдет определенное событие Master, приложение берет на себя управление. Кроме того, часть пользовательских объектов может быть синхронизирована по событиям Master. Например, PI Client настроен на доставку данных по определенному событию.

Регистратор событий предоставляет возможность отслеживать внутренние события Master для инструмента удаленной регистрации. Он хранит предварительно настроенные события Master в переменной PI. Таким образом, инструмент удаленного мониторинга может затем использовать PI logger для сбора данных отслеживания события.

Сетевой драйвер EtherCAT: нулевая копия

В Master 2 обновлен интерфейс сетевого драйвера EtherCAT. В Master 1 при отправке фрейма Master копирует данные в свой промежуточный буфер, а затем драйвер отправляет их. Master 2 же отправляет фрейм напрямую оборудованию без копирования в промежуточный буфер. Master запрашивает буфер у драйвера и передает ему данные. Дрйвер выделяет место в аппаратном пуле (по запросу Master), заполняет его данными фрейма и отправляет их. Этот подход улучшает производительность Master.

Автоконфигуратор

Master 2 позволяет конфигурировать шину в процессе работы. Модуль Автоконфигуратор дает возможность пользовательскому приложению выбрать конфигурацию ведомого устройства (uESI) для применения. Тогда модуль генерирует главный файл конфигурации (ENI) с примененным uESI. В дальнейшем этот ENI будет использоваться при работе Master. uESI — это файл, описывающий конфигурацию пользовательского ведомого устройства, созданный в VS EtherCAT Studio. Можно заранее создать набор uESI и затем применить их к Master в зависимости от потребностей пользователя. Автоконфигуратор можно использовать как минимум двумя способами:

1. Для переключения между различными конфигурациями ведомого устройства.
   Например, сервопривод имеет два рабочих режима: скорость и положение, для которых необходимы разные настройки ведомых устройств. Пользователь может создать два uESI с различной конфигурацией PDO, командами инициализации или другими настройками для каждого режима работы соответственно. Затем пользователь выбирает соответствующий uESI и применяет его в Master через модуль Автоконфигуратор.
2. Для переключения между конфигурациями шины с различным количеством ведомых устройств.
   Master 2 позволяет сопоставлять переменные образа процесса ведомым устройствам по именам ведомых устройств. Благодаря этому приложение ПЛК может задать ведомые устройства для использования. Например, существует два режима работы системы: первый, так называемый Full, предполагает 10 ведомых устройств на шине, второй - Standard - только два ведомых устройства. Использование модуля Автоконфигуратор позволяет настроить Master для работы в режиме Standard. Master сканирует шину, находит только два ведомых устройства, заданных ПЛК, и передает их идентификационную информацию в

Автоконфигуратор. После этого Автоконфигуратор получает uESI ведомых устройств, генерирует соответствующий ENI и отправляет его Master.

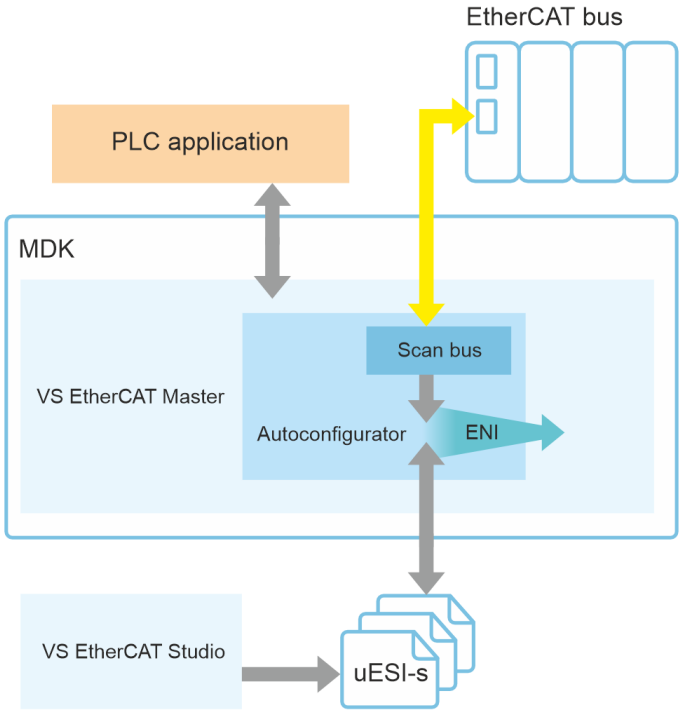


Figure 3. Генерация ENI с помощью Автоконфигуратора

ESC Port Monitor

Master 2 имеет встроенную функцию Port Monitor.
Master автоматически переключает состояние порта ESC с автоматического на автоматическое закрытие, а затем обрабатывает состояние портов (monitors, switches) для защиты циклической связи от неожиданной остановки фрейма.

Аппаратная отправка по времени

VS EtherCAT Master, начиная с версии 2.4, поддерживает функцию Аппаратной отправки по времени. Она позволяет отправлять циклический фрейм точно в начале цикла Master без каких-либо задержек. Обычно Master начинает подготовку циклического фрейма в начале цикла Master. Как результат, фактическое время передачи фрейма задерживается на время подготовки.

Функция Аппаратной отправки по времени использует аппаратный модуль (HW-модуль) на целевой системе и может активироваться только если в целевой системе имеется аппаратный таймер. Если это не так, можно использовать программную эмуляцию. Эмуляция отправки по времени позволяет имитировать функцию отправки по времени. Использование HW-модуля позволяет добиться более высокой точности при отправке фреймов, менее 1 мкс. А также, функция Аппаратной отправки по времени использует планировщик для отправки фоновых фреймов, что ускоряет процесс отправки.

Эмуляция отправки по времени — это программное решение, поэтому оно не обладает преимуществами аппаратного модуля (точностью и скоростью отправки). Эмуляция отправки по времени предназначена для обеспечения того же интерфейса управления, как и Аппаратная отправка по времени. Ее интерфейс позволяет заранее создавать циклические фреймы (автоматически или по запросу пользователя) и запланировать отправку запроса драйверу. Но ввиду программной природы Эмуляции отправки по времени она может вызывать колебание

из-за колебания таймера, зависящего от операционной системы, в то время как Аппаратная отправка по времени не вызывает колебаний, поскольку использует прерывания аппаратного таймера.

Библиотека Интерфейсов Python

Это расширение позволяет взаимодействовать с Master через RPC-сервер (входит в комплект Master по умолчанию). С RPC-сервером можно использовать приложение Python для настройки или диагностики Master. Приложения Python полезны для визуализации различных процессов, создания диаграмм или для Автоконфигуратора.

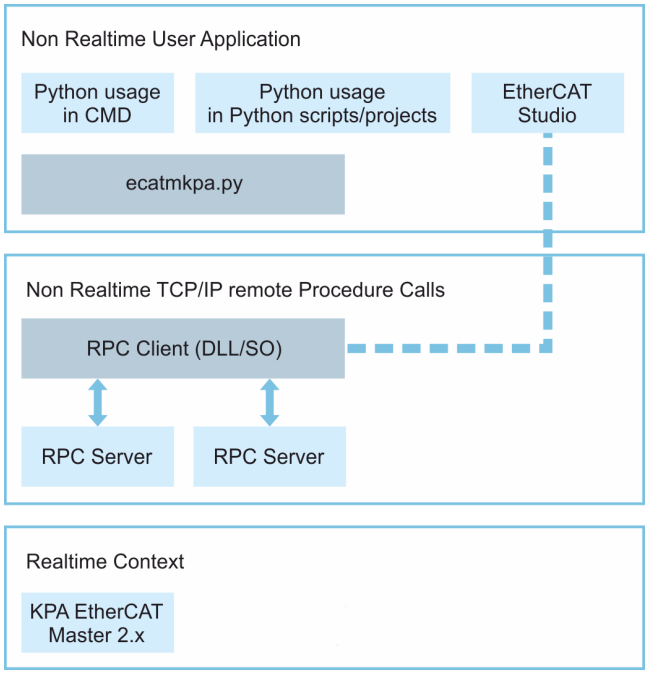


Figure 4. Пример использования Библиотеки Интерфейсов Python

Visutech System Ltd
Klary Tsetkin St, 24-9
220004 Minsk
Belarus
https://visutechsystem.by/

Contact
email: sales@visutechsystem.by
tel.: +375 29 388 70 78

All company processes, from a product order to technical support, are managed according to our quality management system.
Copyright © Visutech System Ltd, Belarus. All rights reserved.
EtherCAT® is registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.