KPA EtherCAT Master 2 Advantages

A new generation of KPA EtherCAT Master (Master) core brings exceptional possibilities of the automation control over EtherCAT network. Master 1 has many limitations due to its inflexible and monolithic architecture. It leads to difficulties in its utilization on some targets, for example:

- Targets with no OS installed
- Targets with limitation of threads number in one process
- Targets with a lack of system resources

In the new Master core we revised its architecture and do our best to avoid all limitations of previous versions and improve and extend Master functionality. Nevertheless, Master 2 keeps compatibility with API layer of Master 1. Here are descriptions of Master 2 features.

Asynchronous control

A new control model is implemented in Master 2 called "asynchronous control" that allows parallel execution of Master's and user's operations and other features based on it.

Parallel operations execution. The asynchronous model makes it possible not to interrupt user execution thread while the Master performs API call. The user can continue the work and check for API call result when it is necessary.

Postponed operations. Asynchronous mode assumes no immediate result, Master executes the operation as soon as it is ready; returned code and data are stored in the operation's handle. Depending on a situation user can either wait for the operation's result or continue working and check if the operation is complete when the user needs it.

Multiple operations. The model allows to execute several operations asynchronously, and to wait for completion of all of them. Asynchronous control requests the usage of a unique identifier (of transactions, sessions etc.) that directly links pending request and operation result. Master uses the object handle as such identifier. All asynchronous operations are always bound to a certain instance of object. It allows getting the current state of a request and extracting an operation result.

Synchronization queue is used for receiving asynchronous calls processing notification; it helps to combine several operations related to different functions, into the one queue to wait. It is possible to handle the whole Master on the single synchronization queue (running a "single thread" model).

In Master 1 with the synchronous execution of operations the user should wait for API call completion and there is no ability to do any work until API call is finished. The asynchronous model makes it possible not to interrupt user execution thread while the Master performs API call. The user can continue the work and check for API call result when it is necessary.

Master Simple Tasks execution

Master 2 implies the task based model that means division of the operations into Master Simple Tasks (MST). In this case MST is the smallest item of execution, i.e. it cannot be stopped. Here is the list of features based on this model.

Master Simple Task execution. It takes minimum execution time and allows receiving a quick reaction on the request.

Simulated multi-thread mode. Dividing the operations into MST-s it is possible to execute them parallel that simulates multi-thread Master mode. The order of MST-s execution is defined by their priorities.

Multithreading. Master 2 allows using any number of Master threads for MST-s execution. In this case all MST-s are assigned to four executors depending on their purpose: real-time, background, synchronization and system. Then executors are assigned to different threads with different priority. In comparison to previous Master versions where the only three threads may be used, such approach brings the following advantages:

> **Distribution of cores load**. Multithreading allows optimal distribution of cores load by assigning to them different MST-s execution.
>
> **Simultaneous tasks execution**. MST-s assigned to different threads can be executed simultaneously.

Multiple EtherCAT Cyclic Tasks

Besides Master Simple Tasks, Master 2 supports multiple EtherCAT Cyclic Tasks (ECT). KPA EtherCAT Studio provides an possibility to generate ECT. These tasks may be controlled by Master or by an external application. Each ECT is assigned to certain Process Data and has particular cycle of execution.

Using ECT-s the user can run them with different cycles, at different times. That allows to control the bus load by assigning the operations with different cycles of execution to ECT-s.

Modular design

Modular design brings an incredibly flexible and adjustable Master core. Almost all parts, including core components, are designed as separate modules. It allows building Master core according to the user’s needs and achievement of optimal performance and functionality balance. Here is the list of Modular design advantages:

> **Split functionality to isolated modules**. Each module is designed as an isolated unit with its own API, PI variables, events, configuration via INI file, and documentation.
>
> **Functionality extension**. A new module for each new functionality should be created. Optimization. It is also easy to optimize Master core by disabling modules or their replacement.

PI-driven control

Master 2 implements new approach to access and work with data in process image (PI). Access to PI is performed by PI client. It is an object that provides read/write access to PI, delivers data from PI to internal client’s buffer, can be subscribed to events and PI changes. New interface allows several users get access to PI simultaneously. PI-driven control brings the list of advantages described below.

Protected access to PI. Each part of PI can be owned by only one client, that restricts a write access to PI. Only one client can write data to specified PI area, it is forbidden for all others.

Multiple PI clients. The user can create any number of PI clients. Each of them will be mapped to particular PI area, subscribed to different events, have different rights and isolated internal buffer.

Mapping of PI variables to the user’s data structure. The user can map particular PI variables to their own structure and then work only with their structure without direct call to PI variables.

Atomic operations. PI client data is processed by single non interruptable task. That guarantees the data consistency.

Multiple delivery buffers. It is a mode when PI client uses several buffers while performing an operation. For example, when Master has already performed the operation but user's application is still busy and cannot fetch the operation's data. In this case PI client can use several delivery buffers and store the data got on each iteration to different buffers. When the user's application becomes free, it consistently fetches the got data from the buffers. Therefore, it guarantees safekeeping of all obtained data.

Delivery by write access or by event. Thanks asynchronous interface and PI-driven control it is not necessary user's application to ask Master data every Master cycle. Now user can once configure Master to send certain (required) data one of the ways:

by write access - when write access to certain area in PI occurs.

by event - certain event occurs.

Diagnostics in Process Image

In comparison to Master 1, in Master 2 all diagnostic data are located in Process Image. This approach allows getting diagnostic data simultaneously with process data. It does not require additional calls for obtaining the diagnostics that simplify Master work.

Now the user can easy watch how changes of Process Data influence the diagnostics changes. To visualize diagnostics changes it is possible to get snapshots of the corresponding signals and view them in the Run-time Data Logger tool in KPA EtherCAT Studio.

PI Logger

Using multiple PI clients now it is possible to monitor changes of any PI variables in run-time mode with the help of Run-time Data Logger tool of KPA EtherCAT Studio. Master captures snapshots of configured PI variables (user's, Slave's or Master service's etc.) on desired event, then stores multiple snapshots in a buffer and send it to Studio. The user can analyze changes of the variables by using Run-time Data Logger tool (in Studio) where they are represented in the chart view.

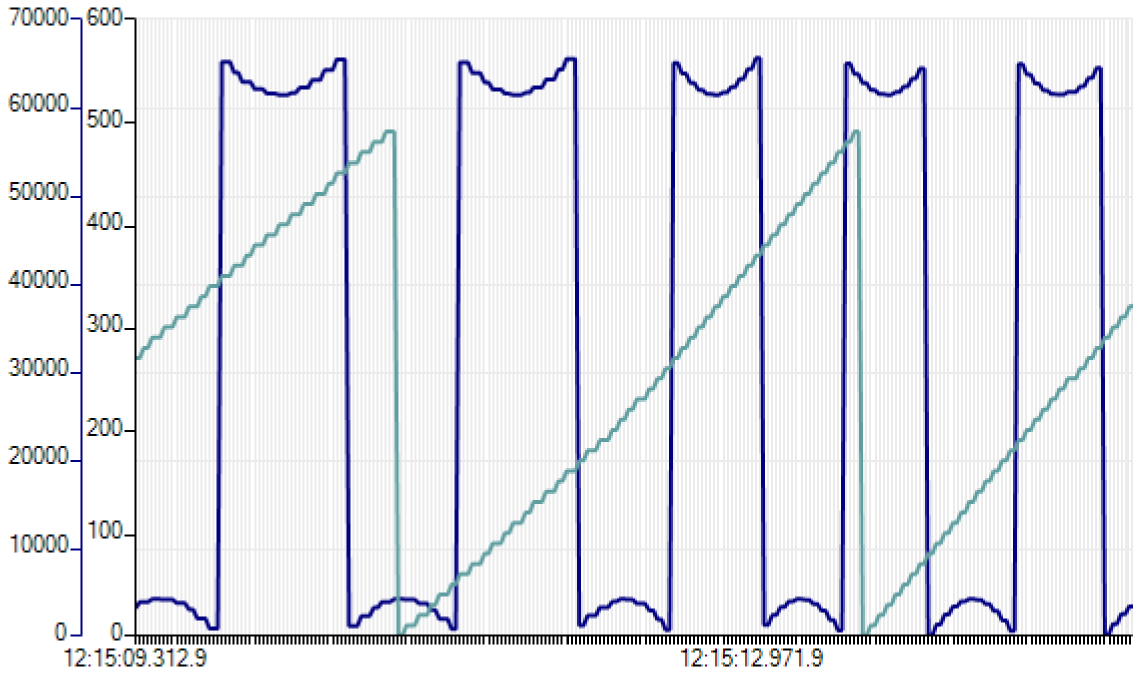

Figure 1. Example of variables chart in Run-time Data Logger

PI Logger can be used as a provider data to third-party tools such as MATLAB or LabVIEW or to any

Python based tools for further advanced analysis.

Events

In comparison to Master 1 where events were used within the Event Handler feature (optional and need to be licensed additionally), new Master 2 core is based on events model. All Master processes are activated/performed by events. Each Master module has events which can be used by both Master and a control application.

Event is a type of message that Master generates in some points of execution, when it is necessary to inform a client about some changes in Master state, configuration, or processing. Events can notify about errors, warnings or just inform about something. All events are generated in the identical format.

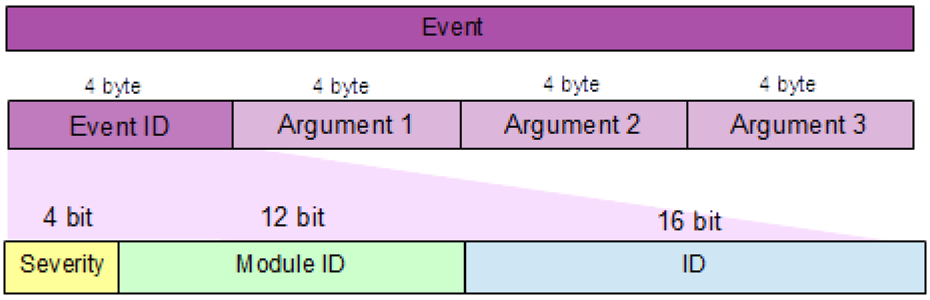

Figure 2. Event format

Where
Severity – severity of Event, allowed values: INFO, WARNING, ERROR or TRACE (can be used for user's events).
Module ID – identifier of Master module which generates corresponding Event notification, e.g. Master module for diagnosing the bus topology (the bus state, connection of devices etc.), or slaves (their current and requested states, failed init commands etc.), or Master (its state, invalid configuration, broken connections etc.).
ID – event identifier in the Master module.
Argument 1, Argument 2, Argument 3 - event specific parameters.

Events are used by Master to synchronize tasks execution, notify about cyclic events, status events or emergency events. All Master events are available on the level of user application. It allows the application to implement the mode of event-driven execution. Most of the time the application sleeps and does not require processor's resources. As soon as the certain Master event happens, the application takes the control. Besides, a part of user-objects may be synchronized by Master events. For example, PI Client is configured to delivery data by a particular event.

Event logger provides ability to sniffer internal Master events for remote logging tool. It stores preconfigured Master events into PI variable. So, remote monitor tool can then use PI logger to capture event trace.

EtherCAT network driver: zero copy

In Master 2 the interface of EtherCAT network driver has been updated. In Master 1 while sending frame, Master copies data to its intermediate buffer and then the driver sends it. Now Master 2 sends frame directly to the hardware without a copying to an intermediate buffer. Master requests a buffer from the driver and passes the data to it. The driver allocates an area in the hardware pool (by Master request), fills it with frame's data and sends it. This approach improves Master's performance.

Master Redundancy

EtherCAT configuration with Master Redundancy enabled comprises one active (primary) Master device and one or several passive (secondary) Masters. Primary Master might not be configured to Master Redundancy, but it is preferable to use KPA EtherCAT Master software to utilize all the advantages of this technology. Secondary Master is connected to the bus as a shadow agent. It is sniffing data telegrams as they pass by without any changes. At the same time, this Master calculates the time when each telegram arrives and tracks possible delays between expected and actual time of arrival.

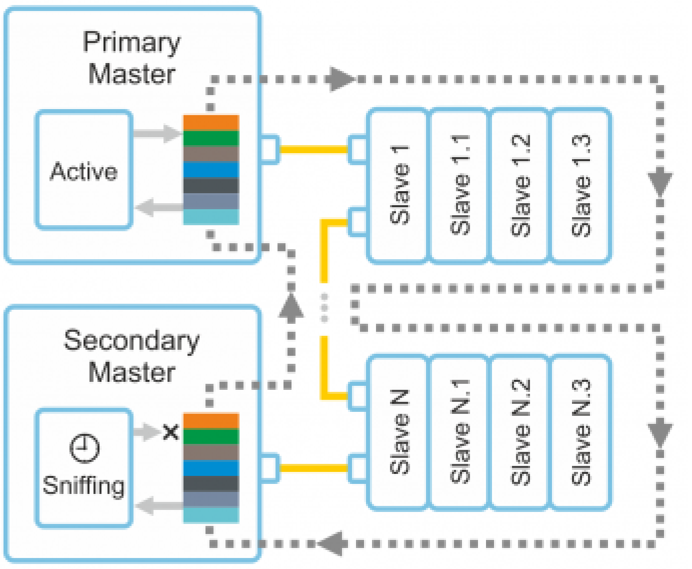

Figure 3. EtherCAT configuration with Master Redundancy

When the telegram is delayed, a watchdog timer starts. After the time specified in the Master's settings is elapsed, the Master starts his failover protocol:

1. Internal switch of the Master is triggered: now the Master is able to write the bus.
2. A new telegram prepared by the secondary Master is written to the bus.
3. In this telegram, a request to free the bus is sent to the active Master, because he might be still online but suffering from some internal troubles.
4. Control application of the new bus Master is notified about going online.

It is possible to return the active Master into its passive state with the bus configurator tool; another passive Master will take control automatically.

To enable multiple redundant Masters, their watchdog timers are configured differently. It is possible to force random generation of the watchdog time for each Master. Thus, we avoid the situation of collision when two or more secondary Masters can attempt to start the failover protocol.

The feature that we propose can protect industrial communication network from a severe, hard to recover failure of its control node. It utilizes architecture peculiarities of EtherCAT technology to implement cost-effective solution that greatly increases fault tolerance without compromising its flexibility and performance. Almost any EtherCAT-enabled industrial automation system can be upgraded with this feature: you need to attach to the bus one or several clones of your controller with only slight or even no modifications of its logic.

Autoconfigurator

Master 2 provides a possibility to configure the bus on the fly. The Autoconfigurator module allows the user application to select slave's configuration (uESI) that will be applied. Then the module generates Master configuration file (ENI) with applied uESI. Further, this ENI will be used at Master work. uESI is a file describing custom slave's configuration and generated in KPA EtherCAT Studio. It is possible to create a set of uESI-s in advance and then applied them to Master depending on the user's needs. The Autoconfigurator functionality may be used at least in two ways:

1. To switch between different configurations of the slave.
   For example, Servo Drive has two working modes Velocity and Position that require different slave settings. The user can create two uESI-s with different PDO-s configuration, Init commands or other settings for each working mode respectively. And then the user selects the corresponding uESI and applies it in Master via the Autoconfigurator module.
2. To switch between bus configurations with different number of slaves.
   Master 2 allows mapping Process Image variables to slaves by slave's names. Due to that the PLC application can set the slaves that will be used. For example, there are two working modes of the system: the first, socalled Full, involves 10 slaves on the bus, the second - Standard - only two slaves. Using the Online Configurator module it is possible to configure Master to work in Standard mode. Master scans the bus, finds only two slaves set by PLC and pass their identification information to Online Configurator. After that, Autoconfigurator gets slaves' uESI-s, generates the corresponding ENI and sends it to Master.

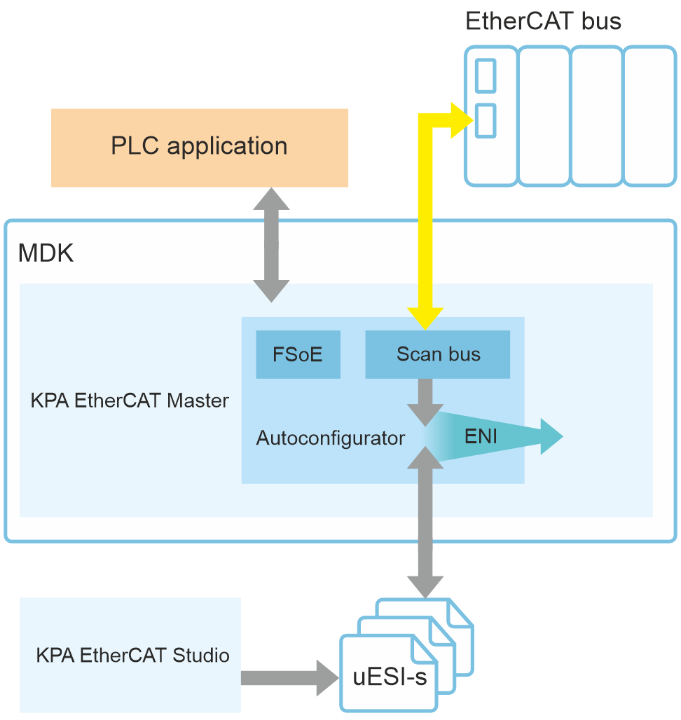

Figure 4. ENI generation via Autoconfigurator

ESC Port Monitor

Master 2 has an inbuilt Port Monitor function.
Master automatically switches ESC port state from Auto to Auto-close and then handles ports status (monitors, switches) to protect cyclic communication from unexpected frame brake.

Hardware Timed Send

The KPA EtherCAT Master starting with version 2.4 supports the function Hardware timed send. It enables the cyclic frame to be sent exactly at the beginning of the Master cycle without any delays. Usually, the Master starts preparing the cyclic frame at the beginning of the Master cycle. As a result, the actual time of frame transmission is delayed by the time of preparation.

The Hardware timed send function uses a hardware module (HW module) on the target and can only be activated if the target system has a hardware timer. If it does not, a software emulation of it can be used. Timed send emulation makes it possible to imitate the timed send functionality. Using the HW module allows to achieve higher accuracy at sending frames, less then 1 us. And also, the Hardware timed send functionality uses a scheduler for sending background frames that speed up a process sending.

The timed send emulation is a software solution so it does not offer the benefits of HW module (accuracy and sending speed). The timed send emulation intends to provide the same control interface as the hardware timed send functionality. Its interface allows to build cyclic frames in advance (automatically or by user request) and to schedule a send request to the driver. But as a software nature of the timed send emulation it can jitter due to an operation system dependent timer jitter, while the hardware timed send does not have a jitter because of using hardware timer interrupts.

Python Interface Library

This extension allows to communicate with Master via RPC Server (included into Master package by default). With RPC Server, it is possible to use Python application to configure or diagnose Master. Python applications are useful to visualize some processes, to create charts/diagrams or for online configuring (the Autoconfigurator feature).

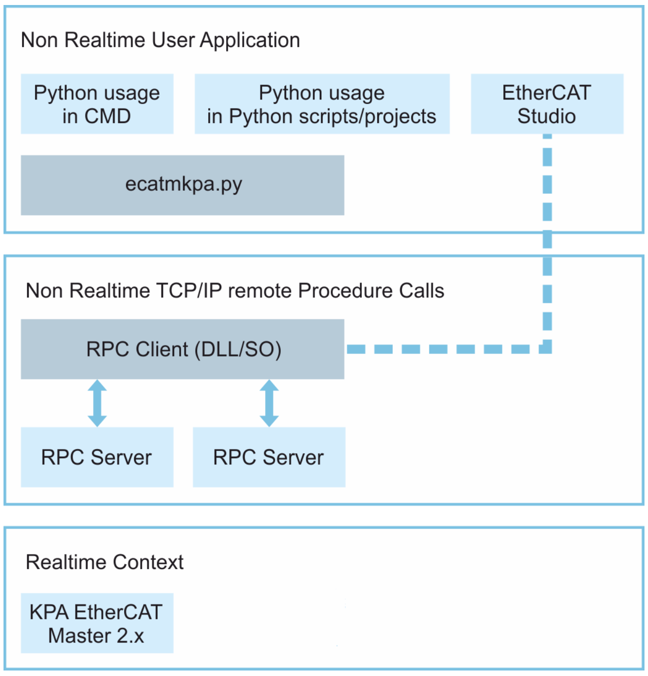

Figure 5. Exampe of Python Interface Library usage