Application Note

EtherCAT Master Architecture

Applicable Products:

Yaskawa Servodrives with CANopen over EtherCAT

Product: Yaskawa Servodrives with CANopen over EtherCAT	**Doc#:** AN.MTN.01
Title: EtherCAT Master Architecture	

Table of Contents

Product: Yaskawa Servodrives with CANopen over EtherCAT	**Doc#:** AN.MTN.01
Title: EtherCAT Master Architecture	

1. About This Document

This document is intended for EtherCAT Master Developers in the planning stages of EtherCAT master development.

This document guides developers in determining architectural features to implement, including:

- Synchronization
  - Distributed Clock reference
- Cyclic update rate
  - Oversampling
- Fault tolerance
- PDO map optimization

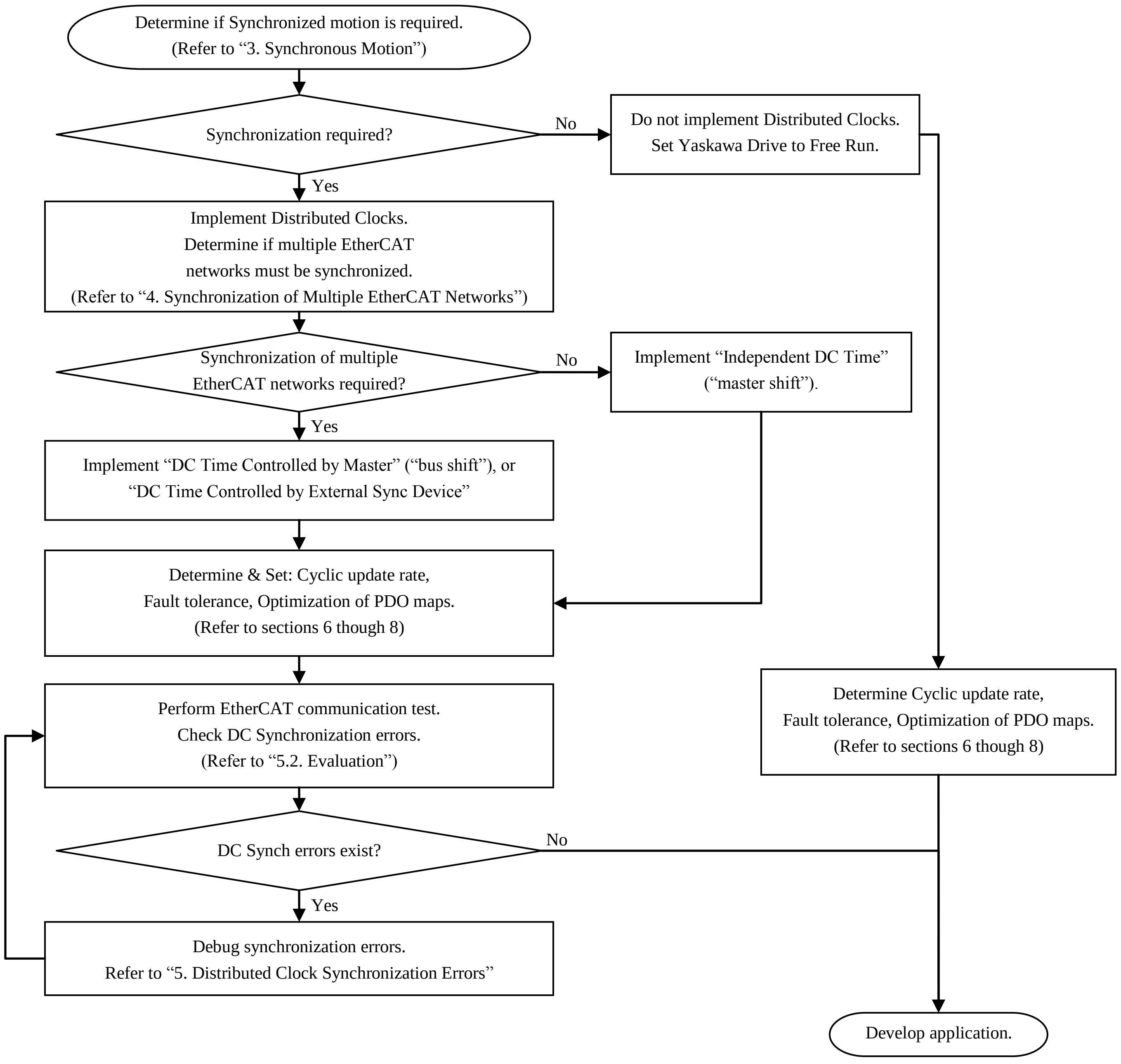
Abbreviations

CoE	- CANopen over EtherCAT	- *Profile for motion control*
DC	- Distributed Clocks	- *Synchronization mechanism*
ESC	- EtherCAT Slave Controller	- *Interface between EtherCAT bus and slave application*
EMI	- Electromagnetic Interference	- *Disturbance affecting EtherCAT communications*
OS	- Operating System	- *Examples: Windows 7, Linux, Mac OS X*
PDO	- Process Data Object	- *Parameter mapped as cyclic communications data*

2. Master Design Features

The decision to implement various features depends on application requirements.

The following flowchart is a guide to determine features to implement.



Determine if Synchronized motion is required.
(Refer to "3. Synchronous Motion")
Synchronization required?
No
Do not implement Distributed Clocks.
Set Yaskawa Drive to Free Run.
Yes
Implement Distributed Clocks.
Determine if multiple EtherCAT
networks must be synchronized.
(Refer to "4. Synchronization of Multiple EtherCAT Networks")
Synchronization of multiple
EtherCAT networks required?
No
Implement "Independent DC Time"
("master shift").
Yes
Implement "DC Time Controlled by Master" ("bus shift"), or
"DC Time Controlled by External Sync Device"
Determine & Set: Cyclic update rate,
Fault tolerance, Optimization of PDO maps.
(Refer to sections 6 though 8)
Determine Cyclic update rate,
Fault tolerance, Optimization of PDO maps.
(Refer to sections 6 though 8)
Perform EtherCAT communication test.
Check DC Synchronization errors.
(Refer to "5.2. Evaluation")
DC Synch errors exist?
No
Yes
Debug synchronization errors.
Refer to "5. Distributed Clock Synchronization Errors"
Develop application.

3. Synchronous Motion

2 design choices exist regarding synchronous motion:

- Implement cyclic motion support (synchronous motion).
  - Points on the trajectory are transmitted each communication cycle
- Implement acyclic motion support (asynchronous motion), examples:
  - The trajectory is transmitted as one command in one communication cycle.
  - The trajectory is written to a reference input buffer table.

3.1. Cyclic motion support

Synchronization is required to support cyclic motion.

Cyclic motion support is necessary if realtime interaction with slaves is necessary.

Examples:

- Coordinated motion – synchronizing position of multiple axes
  - Example: X-Y table tracing a circle
- Time-critical events to occur based on another event.
  - Example: Set an output when motor passes a specific position.

Cyclic motion support allows operation in every Yaskawa servo drive motion mode:

- Profile modes (Position, Velocity, Torque)
- Cyclic modes (Position, Velocity, Torque)
- Interpolated Position mode – Mode 1 & Mode 2
- Homing mode

Distributed Clocks (DC) is the synchronization mechanism for EtherCAT.

Distributed Clocks must be implemented on the EtherCAT master for cyclic motion support.

The Yaskawa Drive must be set to “DC mode”, by setting ESC register 0x980 = 0x0300.

In this mode, the Yaskawa drive can be synchronized to the EtherCAT master with the Sync0 event.

3.2. Acyclic motion support

Synchronization is not required to support acyclic motion.

Acyclic motion support is limited to operation in the following modes:

- Profile modes (Position, Velocity, Torque)
- Interpolated Position mode – Mode 2
- Homing mode

Distributed Clocks do not need to be implemented on the EtherCAT master for acyclic motion support.

The Yaskawa Drive may be set to “Free-Run”, by setting ESC register 0x980 = 0x0000.

In this mode, the cycle during which the drive samples physical inputs and sets physical outputs is not synchronized with the network communication cycle.

4. Synchronization of Multiple EtherCAT Networks

Distributed Clock synchronization requires that all DC devices (slaves and master) be synchronized to one reference clock that is on the network.

3 options for reference clock selection exist for DC synchronization:

- **Independent DC Time** – The DC reference clock is one of the slave devices (typically the first slave on the network that supports DC).
  - acontis terms this as "master shift mode"
  - This term is also referred to as "DC Synchronization to 1st slave clock reference"
- **DC Time controlled by master** – The DC reference clock is the master's clock
  - acontis terms this as "bus shift mode"
  - This term is also referred to as "DC Synchronization to Master Clock Reference"
- **DC Time controlled by External Sync Device** – The DC reference clock is an external synchronization device

4.1. Independent DC Time

This mode is sufficient to operate a single EtherCAT network.

This mode is not sufficient to synchronize multiple EtherCAT networks.

Independent DC Time is the typical mode in systems that require DC.

- This implies that most EtherCAT stacks operate and support this mode without issues.
- Slave devices typically have a clock that is more reliable than a master's clock, which may be affected by load on the master.
- Implementing other DC modes typically involves more development effort.

4.2. **DC Time Controlled By Master**

This mode is sufficient to operate a single EtherCAT network or to synchronize multiple EtherCAT networks.

The master’s clock must be able to operate with the necessary precision to be the EtherCAT network’s DC reference clock.

This mode is one option to implement if multiple EtherCAT networks must be synchronized.

The other option is “DC Time controlled by External Sync Device”.

4.3. **DC Time Controlled By External Sync Device**

This mode is sufficient to operate a single EtherCAT network or to synchronize multiple EtherCAT networks.

This is the least typical mode selected to be implemented in systems that require DC.

An external clock device must exist in the network to act as the reference clock.

This mode is one option to implement if multiple EtherCAT networks must be synchronized.

The other option is “DC Time controlled by master”.

5. Distributed Clock Synchronization Errors

5.1. **Causes**

DC synchronization errors may be caused may various factors.

- DC configuration or implementation errors:
  - Master DC settings not configured properly.
  - Master stack does not conform to EtherCAT specification for DC.
  - Slave DC settings not configured properly.
  - Slave stack does not conform to EtherCAT specifications for DC.
- External factors affecting DC synchronization:
  - Precision of processor clock on master falls outside of EtherCAT requirements for DC.
  - Electrical noise affecting EtherCAT communications.

5.2. **Evaluation**

DC synchronization errors must be evaluated to verify that the DC mechanisms on both the master and slave side are operating properly.

Evaluation methods of the DC synchronization errors include:

- Distributed Clock synchronization errors may be available to evaluate using the master stack API. Check the API documentation for the master stack for this functionality.
- The Yaskawa drive’s mechanism to count missed synchronization events may be monitored to evaluate DC synchronization errors.
  - Refer to “7.3. Monitoring Reception Errors – Slave”.

5.3. **Investigative Actions**

If DC synchronization errors exist, check the following:

- Check that the hardware specifications meet the EtherCAT master stack's requirements.
    - Example: The acontis stack may require a RealTek NIC, of a specific chip number.
- Check that the master computer hardware and OS performance is sufficient for the update rate selected.
    - A Real-Time OS is generally required.
    - Verify SM event and Sync0 jitter. If expected values are exceeded, consult EtherCAT master stack vendor.
- Check & correct the DC settings in the EtherCAT master code
- Check that the network hardware is protected against EMI from nearby equipment, and that the hardware is installed according to the manufacturer's instructions for minimizing susceptibility to EMI (grounding and power filtering details, use of shielded cables, etc).
    - Refer to "7.4. Minimizing Susceptibility to Noise – Slave".
- Check using Yaskawa SigmaWin, the trace including "Position Speed Reference", with a logging rate equal to or higher than the EtherCAT cyclic update rate.
    - Refer to Yaskawa.com document number AN.MTN.03, "EtherCAT Servo Drive Quick Start Guide", section "Verify Motion" for procedure and expected plot output.
- Compare Wireshark traces of the following (or send trace to master stack vendor):
    - EtherCAT master with Yaskawa drive during EtherCAT startup and during DC synchronization errors
    - Beckhoff TwinCAT master with Yaskawa drive during EtherCAT startup.
        - Set up DC the same as the desired DC method for developing EtherCAT master
        - See Yaskawa.com document number, "ENG.09.126" for details on setting up a Beckhoff TwinCAT master with Yaskawa drives.

If DC synchronization errors exist when using "DC Time controlled by master" or "DC Time controlled by External Sync Device", contact Yaskawa Technical Support.
Provide the following information:

- EtherCAT master stack name
- EtherCAT master stack version

6. Cyclic Update Rate

The network's cyclic update rate is defined as the frequency of the master transmitting process data to the slaves.

The cycle time or communication cycle can be determined from the cyclic update rate.

Example: A network's cyclic update rate of 4 kHz has a cycle time of 250 us, and a communication cycle of 250 us.

The following factors affect the selection of a cyclic update rate that meets application requirements:

- Process requirements
  - Example: Closing the position loop in the master.
- Oversampling
  - Example: Setting the cycle time to twice the required rate to counteract missed data
- Cycle time of other devices on the network and features the master supports
  - Example: Yaskawa drive operates at cycle times that are a multiple of 125 us (and up to 4 ms maximum), and another slave that operates at cycle times that are a multiple of 10 us, the master may need to set a lowest common multiple of the two devices (in this case, 250 us).
  - Support of simultaneous varied cycle times is dependent on the master.
  - Refer to the master API documentation for cycle times the master supports.

6.1. Process Requirements

The process requirements dictate the required EtherCAT network cycle time.

Consider the following requirements:

- The rate at which the master requires input process data from the slave (e.g. for data logging).
- Whether or not the master is required to react based on the process data from the slave (e.g. if the master must close a position or velocity control loop through the network).

When possible, designing a system that avoids closing control loops through the network and instead fully utilizes the processing power available on the slave (e.g., by using interpolated position or CSP) minimizes development effort and improves overall system efficiency in the following ways:

- Yaskawa's control and tuning algorithms can be used, providing high-performance motion control with minimum development effort
- Network bandwidth is used more efficiently, allowing a reduced network update to be used
- Network and processing demands on the master are reduced, allowing lower performance hardware to be used without sacrificing system performance

Example: In CoE Cyclic Synchronous Position mode, the master transmits a new position value to the drive each communication cycle. The position values are calculated by the master from a pre-determined path (such as in shape cutting). The next position values are not dependent on the feedback position values from the drive. Velocity is not transmitted by the master in this mode, because the velocity is calculated as the position delta over time (the communication cycle is the time). The tuning algorithms in the drive may be utilized so that the position of the motor matches closely to the position values transmitted by the master. In this application, the master is not required to react based on process data from the slave. The slave is closing the position loop. An acceptable cycle time may be 2 ms in this example.

6.2. Oversampling

Oversampling by setting the cyclic update rate to a faster rate than necessary for the application may counteract missed data.

- The cause for missed data typically is EMI. EMI may be mitigated by installing hardware according to the manufacturer's instructions for minimizing susceptibility to EMI. Refer to "7.4. Minimizing Susceptibility to Noise – Slave".

Example: An application requires that the drive receives position data every 1 ms (1 kHz).
The cycle time is set to 500 us, which results in the master transmitting position data twice in 1 ms.
The drive may miss 1 packet, and still be able to receive updated position data within the requirement of 1 ms. If the drive does not miss any packet, the drive receives 2 packets for every 1 ms.
The cycle time may also be set to 250 us, which results in the drive receiving position data 4 times within 1 ms. The drive may miss 3 consecutive missed packets, and be able to receive updated position data within the requirement of 1ms.

Oversampling may require the application code to perform calculations at a higher rate than without oversampling, and may require a higher class processor.

7. Fault Tolerance

Setting fault tolerance involves the following:

- Setting the allowable communication errors on the master
- Setting the allowable reception errors on the slave

The following procedure is a guideline for setting the allowable errors.

1. Understand the available error counters and monitors on the master and slave.
    - For the Yaskawa drive, refer to "7.2. Setting Allowable Reception Errors – Slave" and "7.3. Monitoring Reception Errors – Slave".
2. Based on the application requirements, determine the minimum required cyclic update rate.
    - Refer to "6. Cyclic Update Rate".
3. Based on the application requirements, determine the maximum number of consecutive allowable communication errors before the master or slave must flag an error that stops the machine operation.
    - Note: Under-specifying this parameter may lead to nuisance flagged errors.
4. Set the allowable communication errors in the master.
    - Refer to "7.1. Setting Allowable Communication Errors – Master".
5. Set the allowable communication errors in the slave.
    - Refer to "7.2. Setting Allowable Reception Errors – Slave".
6. Test in an environment that meets the following:
    - Observes hardware installation guidelines for minimizing susceptibility to noise.
        - Refer to "7.4. Minimizing Susceptibility to Noise – Slave".
    - Matches the real-world operating environment of the application.
7. If errors are flagged, re-evaluate items 2 and 3 above and iterate.

7.1. Setting Allowable Communication Errors – Master

The EtherCAT master may include functionality to monitor transmission and reception errors.
The application code may be written to utilize the error monitors to command the motion to stop or notify the user of communication errors.

7.2. Setting Allowable Reception Errors – Slave

Process data reception errors are monitored as SM2 event miss counts, and may only be monitored if DC is enabled.

- Yaskawa drives: Integrator function utilized for issuing a fault for missed SM2 events.
- Non-Yaskawa drives: Other fault-issuing functionality such as faulting after 2 consecutive SM2 event misses.

Yaskawa's integrator function operates as follows:

- DC must be enabled.
- If an SM2 event was missed, increment the internal error counter by 3.
- If an SM2 event was received, decrement the internal error counter by 1 (counter minimum is 0).
- If the internal error counter is equal to or exceeds the Internal Error Counter Limit (object 10F1:02h), output the fault "EtherCAT Outputs Data Synchronization Error" alarm A.A12 and stop motion.
- The default setting for the internal error counter is 9.
  - Output fault at the 3rd consecutive missed packet.
- The maximum setting for the internal error counter is 15.
  - Output fault at the 5th consecutive missed packet.
- The fault output is disabled by setting the Internal Error Counter Limit to 0.
  - Monitor the external SM2 event miss counter and implement fault behavior in the master.
- The internal error counter increments for an SM2 event miss. This is different from a missed packet. A master may not increment its missed packet counter even though the drive has encountered an SM2 event miss.

7.3. Monitoring Reception Errors – Slave

Firstly, the EtherCAT cycle time must be a multiple of 125us.

- Make this setting with the EtherCAT master.
- SigmaWin motion monitor "1C32h:02:Cycle time" will read as a multiple of 125000.

An external SM2 event miss counter monitor is available with details as follows:

- The monitor is visible though EtherCAT as object 1C32:0Ch, "SM2 event miss count".
- The monitor is visible in the SigmaWin motion monitor.
- DC must be enabled (ESC register 0x980 = 0x0300), set by the EtherCAT master.
    - SigmaWin motion monitor "1C32h:01:Synchronization type" will read "H.0002".
- Increments by 1 for each SM2 event miss.
- Resets to 0 when EtherCAT state changes out of Operational state.
- Operates regardless of 10F1:02h setting (10F1:02 may be set to 0 which disables fault output, but the SM2 event miss counter monitor will still increment normally).

7.4. Minimizing Susceptibility to Noise – Slave

Prior to machine design, refer to Yaskawa.com document number "eng02.016" for noise reduction wiring and grounding techniques & guidelines, as well as the Yaskawa drive's user's manuals. Applications with excessive noise may require additional noise reduction measures.

During machine design, the following technique may be used to minimize susceptibility to noise.

1. Run the machine.
2. Check the rate of increase of master and slave error counters.
3. Make changes to the hardware installation.
4. Check the rate of increase of master and slave error counters.
5. Compare the rates from step 2 and step 4.
    a. A decrease in the rate of increase may indicate a system with improved noise immunity.
    b. An increase in the rate of increase may indicate a system with worse noise immunity.

8. Optimizing Cyclic Data

The number of cyclic data objects is limited depending on the drive specification.

- Example: Yaskawa SGDV CoE drive is limited to 16 objects for TxPDO and 16 objects for RxPDO by reading the PDO information from the drive (rather than reading the ESI for PDO information) and assigning 2 PDO maps to each Sync Manager PDO assignment.

Because the number of objects is limited for cyclic data exchange, optimization may be necessary.

Optimizing the number of cyclic data objects includes the following benefits:

- Meeting the object limit of the drive specification's PDO map & Sync Manager PDO assignments.
- Additional bandwidth becomes available for additional slaves to be added to the network.
- Shorter network cycle time becomes available.
- Master load may decrease.

Depending on the application, objects may be removed from the PDO map, and accessed though SDO communications.

Example: If dynamic mode switching is not required for the application, the object "Modes of Operation" 6060h may removed from the PDO map, and modified though SDO communications.

Conversely, master development may be simplified by filling the PDO map, because access to the objects though PDO communication typically involves less function calls.

Appendix A: **Troubleshooting**

Any Issues

Test with known working master – Does the same behavior exist with TwinCAT? Refer to Yaskawa.com document number AN.MTN.03 "EtherCAT Servo Drive Quick Start Guide".

Update master software – Check with the EtherCAT master stack provider or EtherCAT master vendor for latest updates to firmware and software. Use the latest release from the vendor.

Verify compatibility – If the master has been present at an EtherCAT plugfest, it is less likely to have issues.

Obtain network trace – Consider if a Wireshark trace can provide further troubleshooting details. Hardware designed for EtherCAT Wireshark tracing: Beckhoff ET2000.

- Necessary if nanosecond resolution is required (timestamp is added to packet).

Search ethercat.org for additional information about Wireshark usage.

Issues with Initialization

Regenerate ENI file – If the ENI file was not generated using Beckhoff's EtherCAT Configurator (ET9000) or using TwinCAT, use an ENI file generated by either of those programs.

Issues with Synchronization

Refer to "5. Distributed Clock Synchronization Errors"