EtherCAT timing FAQ No.0005

Part	Version	Revision	Datum	Status
en	6.3.0.0	005	2019-07-23	Released

Content

KEB Automation KG
Südstraße 38
32683 Barntrup, Germany
E-mail: info@keb.de
Tel. +49 5263 401-0
Fax +49 5263 401-116
www.keb.de

District Court Lemgo HRA 5649
DUNS-No. 314108728
VAT-No. DE309087075
Bank Details: Sparkasse Paderborn-Detmold
IBAN DE 19 4765 0130 0000 0060 07
BIC WELADE3L

General Partner:
Vittorio Tavella
KEB Verwaltungs-GmbH, Barntrup
District Court: Lemgo HRB 8965
Directors: Curt Bauer CMO,
Ralf Lutter COO, Vittorio Tavella CFO,
Wolfgang Wiele CTO

Introduction

This document gives a general overview about the KEB EtherCAT timing behaviour for synchronous operations. General terms and possible user settings are explained.

Timing (Frame at task start=false)

General master cycle

The general PLC task is to control a technical process in real-time with deterministic behaviour. Real-time capability is the ability to react in time on process inputs (sensors) by controlling process outputs (actors) to guarantee a stable behaviour. The processdata is exchanged through a fieldbus master hardware. The fieldbus stack handles the communication between master and slave devices. This document describes the processdata exchange based on the EtherCAT fieldbus technology.

A general PLC cycle is structured into certain events and sequences. The cycle is repeated continuously in hard regulated time intervals (**Cycletime**).The cycle start event (event a) jitters depending on the regulation quality (range A). The cycle sequence mainly handles reading of process in data (range B), execution of user code (range C+D) and writing of process out data (range E).

The user code is freely programmable according to IEC61131-3 with KEB COMBIVIS studio 6. Depending on the target hardware performance, every instruction call needs a certain amount of time. Depending on number and type of instructions, the user code execution varies in each cycle between min. cycletime and max. cycletime (range D).

Deterministic behaviour can be achieved, if the max. cycletime is always smaller than the adjusted cycletime.

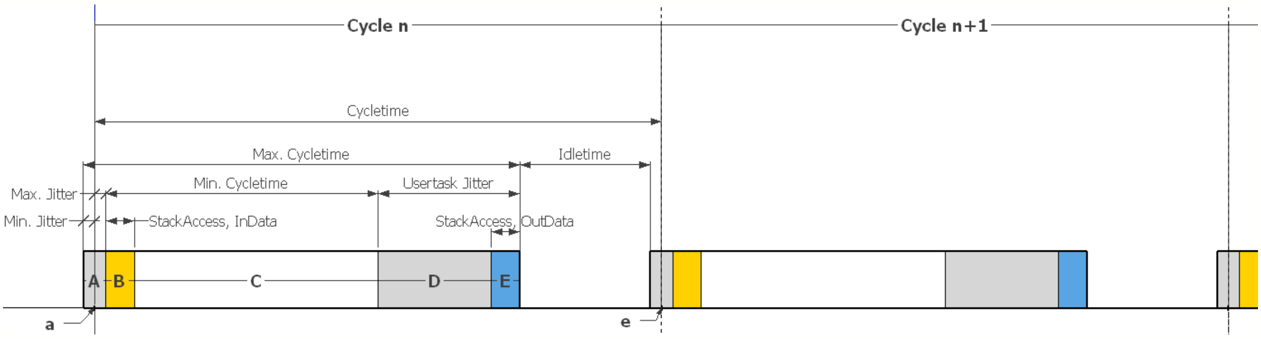


Diagram 1 - General master cycle (ethercat master task)

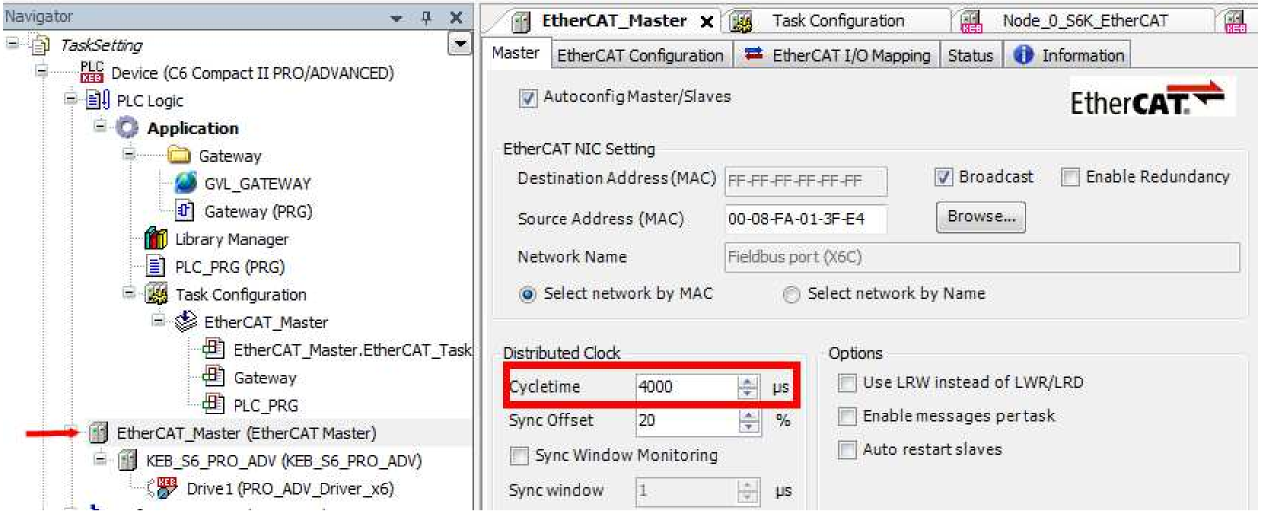
a Average cycle-start event, Execution start of plc cycle n
e Average end-of-cycle n, execution start of plc cycle n+1

A Jitter of PLC task startevent.
B Stack access to read in data.
C User code execution. Interpretation of current process values and calculation of set values to control a technical process.
D Jitter of usercode execution time
E Stack access to write out data and send the ethercat telegram.

Cycletime Configured EtherCAT-Bus Cycle time (Setting of EtherCAT-Master / EtherCAT-Task).
Max. Cycletime: Maximum measured runtime of PLC cycle.
Min. Cycletime: Minimum measured runtime of PLC cycle.
Idletime: Buffer/ Idle time at the end of the PLC task until startevent of next cycle.
Max. Jitter Jitter of PLC task, maximum value.
Min. Jitter Jitter of PLC task, minimum value.
Stack Access, InData Time needed for stack handling and to read process input data (including stack related offset time)
Stack Access, OutData Time needed for stack handling and to write process output data

Where to find these values in a standard project?

Cycletime



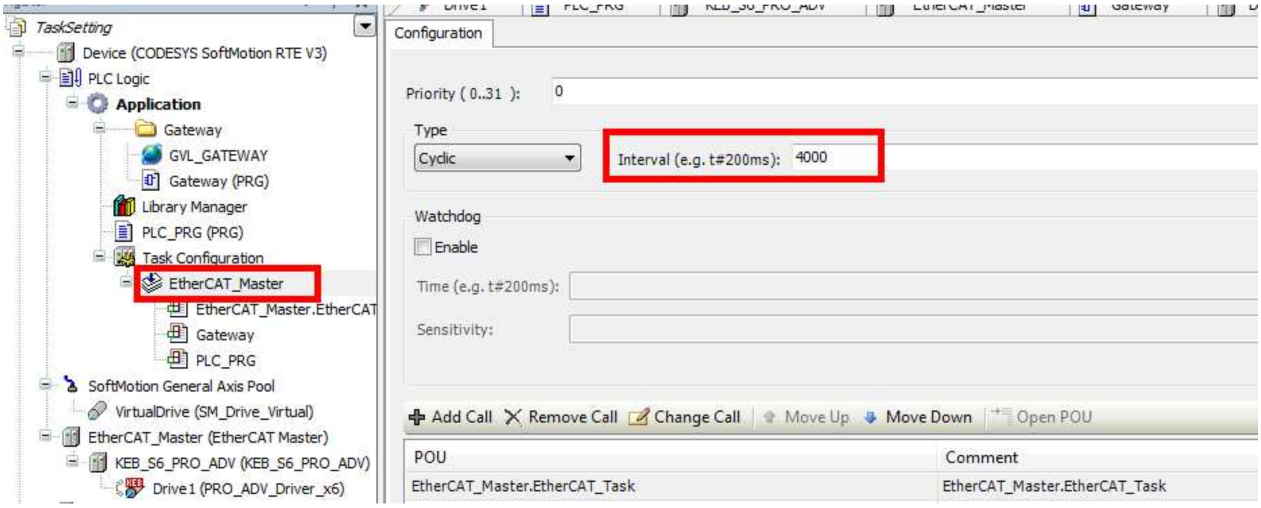
☑ Autoconfig Master/Slaves

EtherCAT NIC Setting
Destination Address (MAC): FF-FF-FF-FF-FF-FF
☑ Broadcast
☐ Enable Redundancy
Source Address (MAC): 00-08-FA-01-3F-E4
Browse...
Network Name: Fieldbus port (X6C)
● Select network by MAC
○ Select network by Name

Distributed Clock
Cycletime: 4000 µs
Sync Offset: 20 %
☐ Sync Window Monitoring
Sync window: 1 µs

Options
☐ Use LRW instead of LWR/LRD
☐ Enable messages per task
☐ Auto restart slaves

Cycletime is an EtherCAT master device setting.

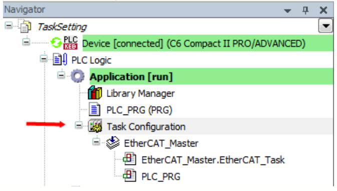


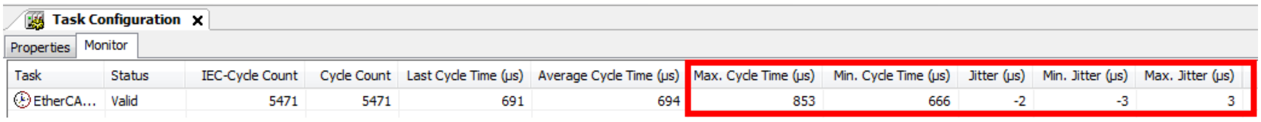
Priority (0..31): 0
Type: Cyclic
Interval (e.g. t#200ms): 4000
Watchdog
☐ Enable
Time (e.g. t#200ms):
Sensitivity:

POU	Comment
EtherCAT_Master.EtherCAT_Task	EtherCAT_Master.EtherCAT_Task

The EtherCAT_master task setting has to be adjusted with the same value.

Jitter
Min. Jitter, Max. Jitter, Max Cycletime, Min. Cycletime are measured and shown by the taskconfiguration → monitor





Task	Status	IEC-Cycle Count	Cycle Count	Last Cycle Time (µs)	Average Cycle Time (µs)	Max. Cycle Time (µs)	Min. Cycle Time (µs)	Jitter (µs)	Min. Jitter (µs)	Max. Jitter (µs)
EtherCA...	Valid	5471	5471	691	694	853	666	-2	-3	3

Send cycle, 2 master cycles

Diagram 2. shows how data is sent from the EtherCAT master device to a slave device. The shown sequence is valid for synchronous operations and master setting “FrameAtStart=false”. For instance data n is calculated in cycle n (range C), It is sent via telegram at the end of the cycle (range E, F) and arrives in the slave fieldbus hardware (range G). After the next synch interrupt (event s) the available data is exchanged with the slave firmware (range H) and computed in the slave specific control sequence (range I). The positon of the telegram arrival (range G) depends on the adjusted cycletime, adjusted synch offset, min./ max. cycletime , further time constants and jitter depending on hardware and used software stack. Furthermore the position of StackAccess (range E) is regulated by the PLC, so it depends on the measured timings of the previous cycles.

To achieve deterministic behaviour, it must be avoided that the range of telegram arrival (range G) overlaps with the range of telegram access (range H).

As the position of the telegram arrival depends on the user application and used hardware, it is strongly recommended to verify the telegram position for each application!

This can be done offline by using the attached calculation formulas, or online by usage of the KEB EtherCAT wizard.

To shift the telegram arrival range a **Synch Offset** can be definied [-50..50% of cycletime]. Positive and negative shifting of the PLC relative to synch event **s** is possible.

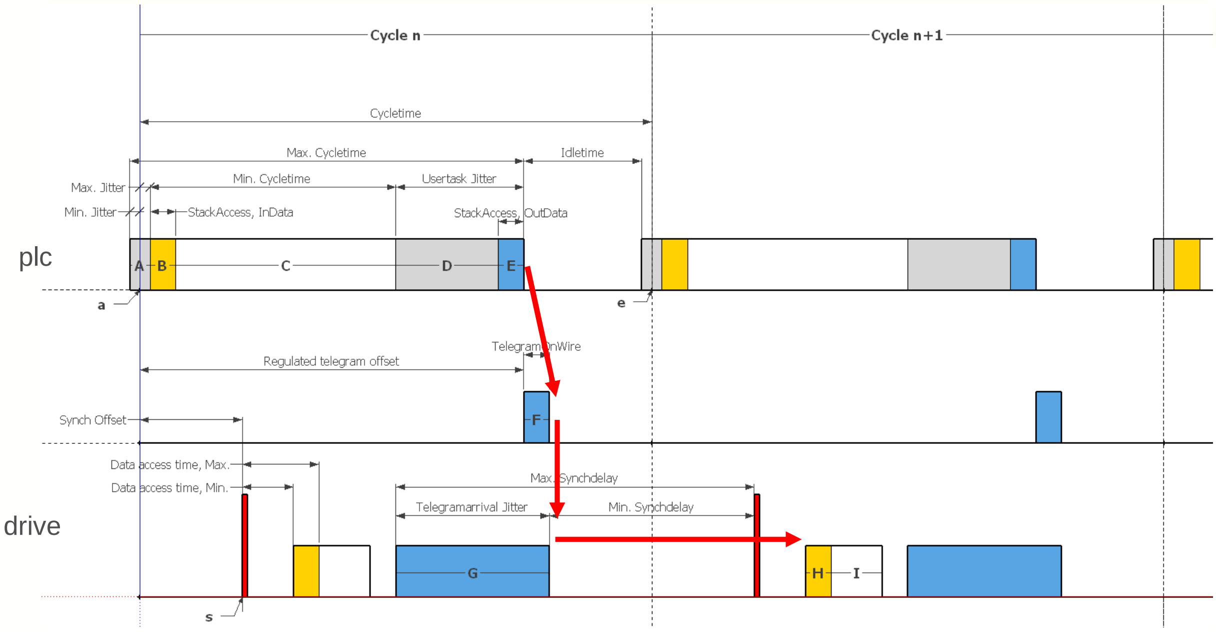


Diagram 2 - telegram send cycle

C: PDO data n is calculated in PLC cycle
F: Data n on wire. The data arrives the slave with the falling edge of this range.
G: Telegram arrival range. The range jitters depending on the master cycle positon and jitter.
H: PDO buffer is read/ written by slave.
I: Data n is processed by slave firmware.

TelegramOnWire (range F) Telegram runtime, typically approx. 50µs

Max. Synchdelay : Falling edge of tTelegramOnWire. related to slave synch event
Min. Synchdelay: Falling edge of tTelegramOnWire. related to slave synch event

Telegramarrival Jitter,
(range G): Jitter of PDO arrival at slave device.

Telegramarrival Jitter : Max. Cycletime- Min Cycletime (plc related calculation.)
Telegramarrival Jitter: Max. Synchdelay - Min Cycletime (drive related calculation)

Synch Offset [%]: Offset of plc-cycle start (relative to sync) in percent of cycletime.
Data access time, Min: Time of EtherCAT slave’s earliest access to process-data (relative to sync). Almost constant factor depending on slave type and firmware.
Data access time, Max: Time of EtherCAT slave’s latest access to process-data (relative to sync). Almost constant factor depending on slave type and firmware.

Hint:

Timings and calculations are valid only for stable cycles. EtherCAT startup sequence and instable applications with cycletime overflows are excluded.

Where to find these values in a standard project?

Synch delay, Telegram arrival jitter
The telegram arrival times related to synch event are measured by KEB drives and shown in the EtherCAT wizard.

Cycletime:	Fb10
Min. synch delay:	Fb27,
Max. synch delay:	Fb28

Hint: Parameter Fb28, Fb27 are showing a positive value, but the negative distance to synch event is measured. To compare it with calculated values, it must be multiplied by -1.

Telegramarrival Jitter: Blue Range:= Fb27-fb28

s Synch event: Red pulse line
Data Access: Yellow-orange time-slot of slave’s data-access.
Tolerance time 1: Buffertime after slave data-access, before telegram arrival
Tolerance time 2: Buffertime after telegram arrival, before next slave data-access

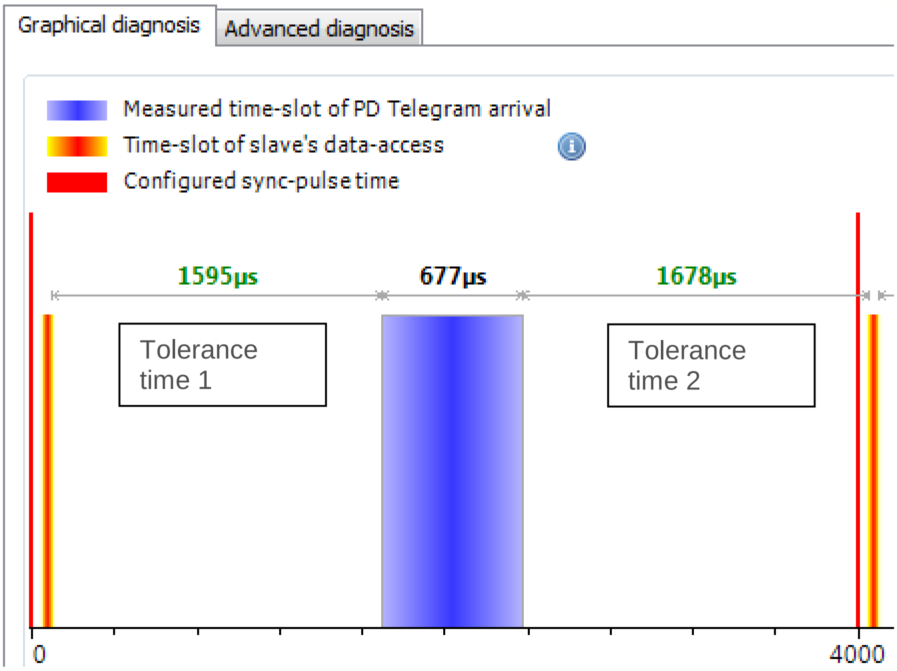
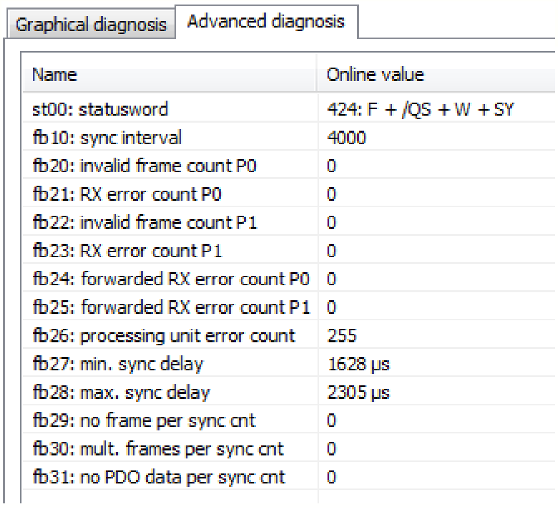
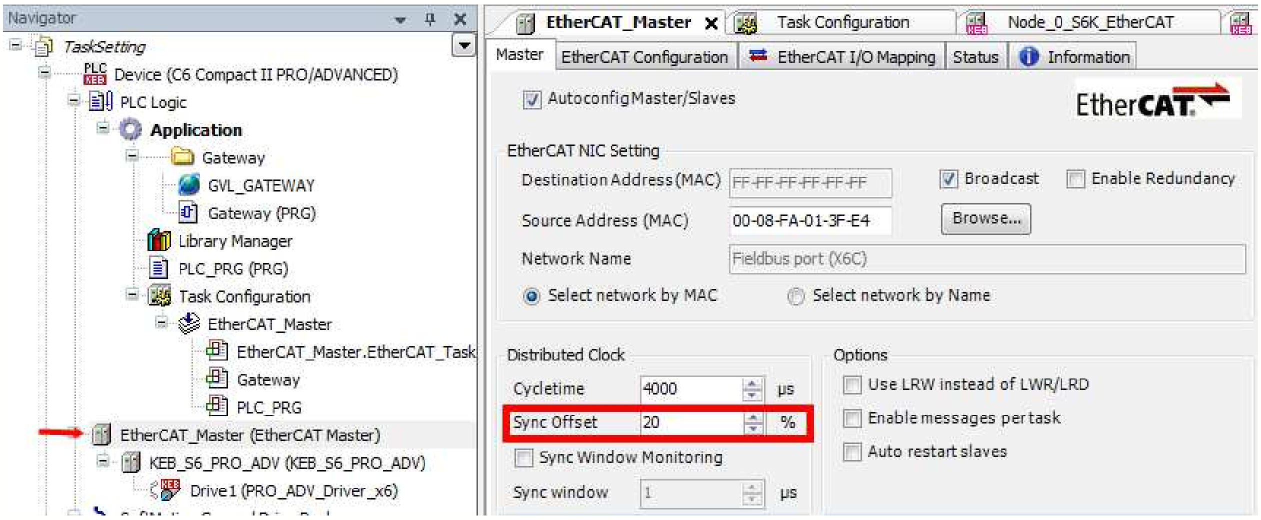

Graphical diagnosis
Advanced diagnosis
Measured time-slot of PD Telegram arrival
Time-slot of slave's data-access
Configured sync-pulse time
1595µs
677µs
1678µs
Tolerance time 1
Tolerance time 2
0
4000

Diagram 3 - EtherCAT wizard



Name	Online value
st00: statusword	424: F + /QS + W + SY
fb10: sync interval	4000
fb20: invalid frame count P0	0
fb21: RX error count P0	0
fb22: invalid frame count P1	0
fb23: RX error count P1	0
fb24: forwarded RX error count P0	0
fb25: forwarded RX error count P1	0
fb26: processing unit error count	255
fb27: min. sync delay	1628 µs
fb28: max. sync delay	2305 µs
fb29: no frame per sync cnt	0
fb30: mult. frames per sync cnt	0
fb31: no PDO data per sync cnt	0

Sync Offset



☑ Autoconfig Master/Slaves

EtherCAT NIC Setting

Destination Address (MAC): FF-FF-FF-FF-FF-FF ☑ Broadcast □ Enable Redundancy

Source Address (MAC): 00-08-FA-01-3F-E4

Network Name: Fieldbus port (X6C)

● Select network by MAC ○ Select network by Name

Distributed Clock

Cycletime: 4000 µs

Sync Offset: 20 %

□ Sync Window Monitoring

Sync window: 1 µs

Options

□ Use LRW instead of LWR/LRD

□ Enable messages per task

□ Auto restart slaves

The synch offset can be adjusted at the EtherCAT master device in a range of -50..50% of the cycletime.

Shift Time

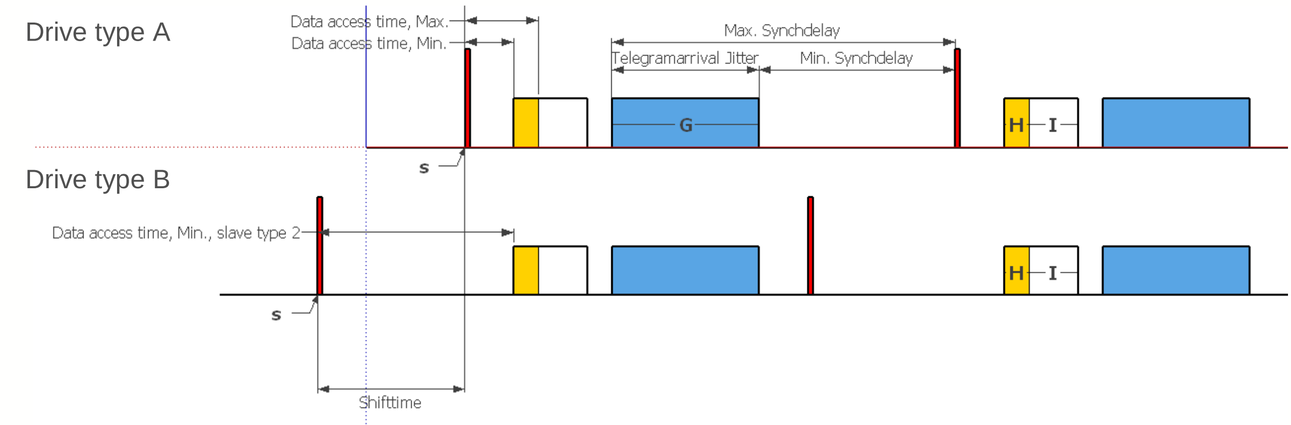
In contrast to the synch offset, which is shifting the master cycle, the slave shif time works on the cycle of each slave individually.

By changing the shift time the synch event of a single slave and so all slave related ranges like the timeslot of slave data access, are shifted. In general the default value given by the vendor device description should be used. In special cases of application the value may be changed.

Application samples

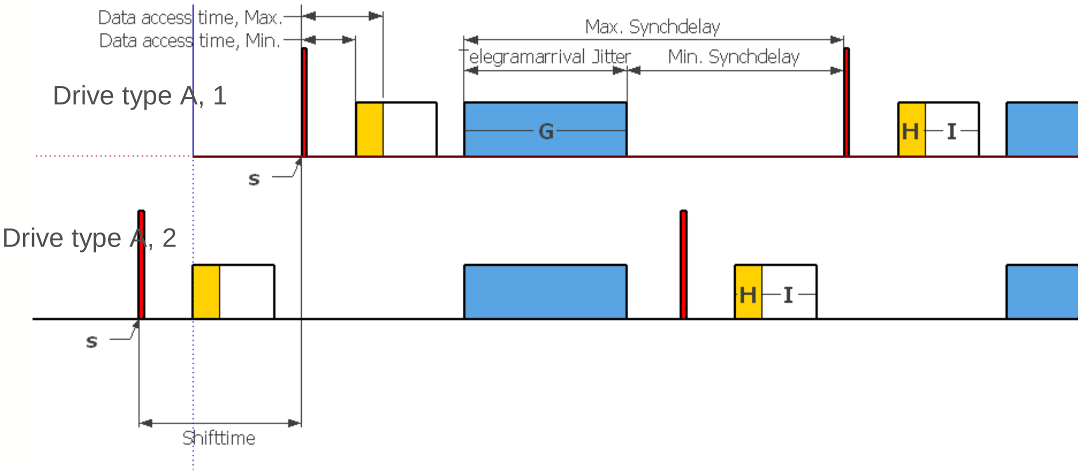
A) Synchronize slave data access of different slave types.

Different slave types may have different data access timings, and so the influence on the technical process may be shifted. The technical process may be synchronized with higher accuracy by synchronizing the slaves data access.

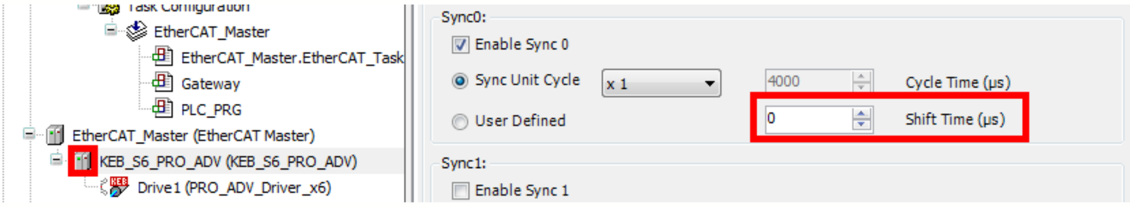


B) Desynchronize slave data access of same slave type.

The synchronized access of several slaves to the technical system may generate negative side effects (e.g. PWM related EMC, voltage drop of DC-circuit). The side effects may be reduced by desynchronization of the slaves data access.



Where to find the Slave Shift Time


Task Configuration
EtherCAT_Master
EtherCAT_Master.EtherCAT_Task
Gateway
PLC_PRG
EtherCAT_Master (EtherCAT Master)
KEB_S6_PRO_ADV (KEB_S6_PRO_ADV)
Drive1 (PRO_ADV_Driver_x6)

Sync0:

☑ Enable Sync 0

● Sync Unit Cycle x 1 — 4000 Cycle Time (µs)

○ User Defined — 0 Shift Time (µs)

Sync1:

☐ Enable Sync 1

Determination and adjustment of telegram position

Calculation

The calculation shows, how to estimate the telegram arrival position **without** using the EtherCAT wizard.

This step can be skipped, if measured data from the EtherCAT wizard is available.
A buffer/tolerance between the position of the telegram arrival, so called synch delay and the position of slave data access is necessary to guarantee synchronous operation. The tolerance time before and after the telegram arrival must be greater than 0. The tolerance times can be calculated by comparing the position of telegram arrival and position of slave data access.
Tolerance time 1 (see diagram 3, ethercat wizard):= Cycletime - Max. Synchdelay- Data access time, Max
Tolerance time 2 (see diagram 3, ethercat wizard):= Data access time, Min - Min. Synchdelay

The synch delays can be estimated by the following formula (According to diagram Diagram 2 - telegram send cycle):
Min. Synchdelay = CycleStart + Min.Cycletime + Min. Jitter + TelegramOnWire - Shifttime
Max. Synchdelay = CycleStart + Max.Cycletime + Max. Jitter + TelegramOnWire - Shifttime

The estimated CycleStart offset *(Event a, diagram 2)* can be calculated by the formula:
*CycleStart = - SynchOffset *Cycletime*

Additionally the width of the telegram arrival (see diagram 3, ethercat wizard) can be calculated by the formula:
Telegramarrival Jitter= |Max. Synchdelay – Min. Synchdelay|

Sample:

SynchOffset:	20%:
Cycletime:	4ms
Data access time, Max (S6K):	150µs
Data access time, Min (S6K):	50µs
TelegramOnWire:	50µs
Max. Cycletime:	581µs
Min. Cycletime:	529µs
Min. Jitter:	-4µs
Max. Jitter:	+4µs
Shifttime (default)	0µs

*CycleStart = - SynchOffset *Cycletime*
*CycleStart := - 0.2*4000µs:= -800 µs(before synch):= 3200µs (after previous synch)*

Max. Synchdelay = CycleStart + Min.Cycletime + Min. Jitter + TelegramOnWire -Shifttime
Max. Synchdelay = -800 µs + 529 *µs + (-4) µs + 50 µs* + 0 *µs = -225µs*

Min. Synchdelay = CycleStart + Max.Cycletime + Max. Jitter + TelegramOnWire - Shifttime
Min. Synchdelay = -800 µs + 581 *µs + 4 µs + 50 µs* + 0 *µs = -165 µs*

Telegramarrival Jitter= |Max. Synchdelay – Min. Synchdelay|= |-225 µs – (-165 µs)| = 60 µs

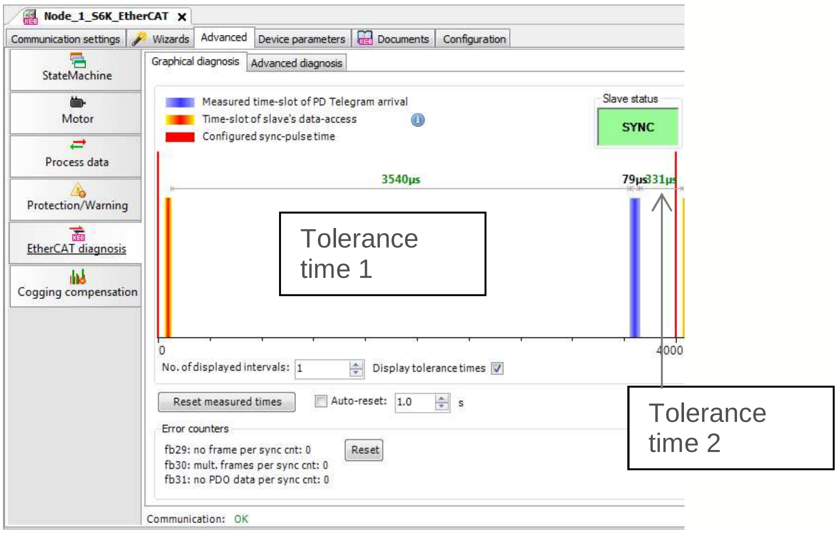
Tolerance time 1 (see diagram 3, ethercat wizard):= Cycletime - *Max. Synchdelay- Data access time,*

Max
Tolerance time 1 = 4000µs – *(-150µs) – 225µs*= ***3625µs***

Tolerance time 2 (see diagram 3, ethercat wizard):= *Data access time, Min - Min. Synchdelay*
Tolerance time 2 = *50µs – -165µs*= ***310 µs***

Interpretation/ Risk of overlapping:
The telegram arrival and range of slave pdo access are not overlapping.
Synchronous operation is possible, if the application jitter stays stable. Tolerance time 2 is low and should be optimized by shifting the telegram arrival range.

Measurement



No. of displayed intervals: 1
Display tolerance times ☑
☐ Auto-reset: 1.0 s

fb29: no frame per sync cnt: 0
fb30: mult. frames per sync cnt: 0
fb31: no PDO data per sync cnt: 0

Communication: OK

Diagram: 4 Measured tolerance times

Fb27: synch offset, Min: 0µs (relative to data access time)
Fb28: synch offset, Max: 3672µs (relative to data access time)

Tolerance time 1, actual (see diagram 4) = ***3540 µs***
Tolerance time 2, actual (see diagram 4):= **331 *µs***
Telegramarrival jitter:= 79µs

Interpretation/ Risk of overlapping:
The range of telegram arrival and range of slave pdo access are not overlapping, the error counters are stable at 0. Synchronous operation is possible, if the application jitter stays stable. Tolerance time 2 is low and should be optimized by shifting the telegram arrival range.

Optimization of tolerance times:
The range of telegram arrival can be optimized, by shifting it into the middle of 2 time-slots of slave's data access.

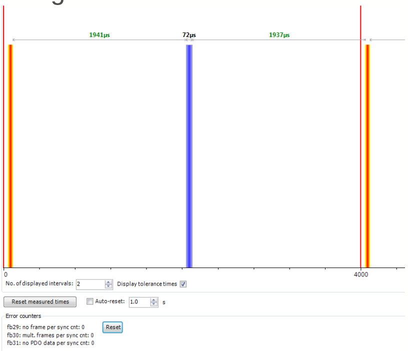
Tolerance time, opt. = (Tolerance time 1 + Tolerance time 2)/2
Tolerance time, opt. = (3540+331)/2 = 1935 µs

SynchOffset, opt.: = SynchOffset, actual + (Tolerance time, opt - Tolerance time, actual)/ Cycletime
SynchOffset, opt: =20% + (1935µs - 331µs)/ 4000µs =20%+40%= 60% (after synch)

The range of SynchOffset. is [-50..+50], therefore the value 60% has to be converted into a negative offset.

SynchOffset, opt.: = 60%-100%= -40% (before synch)

Measured result:
Telegram arrival in the middle of slave pdo access ranges.

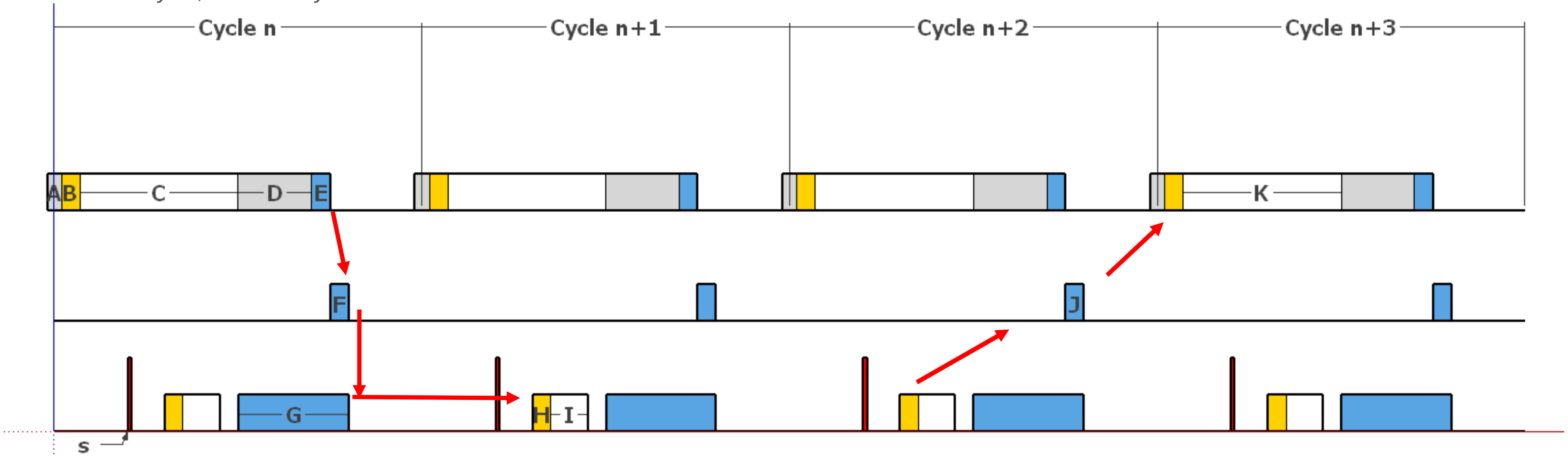


No. of displayed intervals: 2 Display tolerance times ☑

☐ Auto-reset: 1.0 s

Error counters
fb29: no frame per sync cnt: 0
fb30: mult. frames per sync cnt: 0
fb31: no PDO data per sync cnt: 0

Send/ recieve cycle, 4 master cycles



C: PDO data n is calculated in PLC cycle
F: Data n on wire. The data arrives the slave with the falling edge of this range.
G: Telegram arrival range. The range jitters depending on the master cycle positon and jitter.
H: PDO buffer is read/ written by slave.

I: Data n is processed by slave firmware.
J: data n on wire in direction of PLC
K: Data n re-read in PLC

Summary: Frame at task start=false

Advantage:
A write/ read back cycle is reduced to 4 cycles. If the synchoffset is optimized, a fast reaction on calculated values on the slave is possible.

Disadvantage:
The telegram arrival is directly effected by the runtime of the user program (Cycletime, min, Cycletime, max.,Jitter, Min, Jitter, Max). This can put synchronous operation at risk, if the user program execution time is not steady.
It is recommended to calculate or measure the telegram arrival time relative to slave telegram access time for every application separately! The master SynchOffset should be used to shift the telegram arrival into a save range.

Timing (Frame at task start=true)

Send cycle, 2 master cycles

The IoDrvEtherCAT stack beginning from version 3.5.3.50 allows to send the frame at the beginning of the plc cycle (before the IEC user program). This will change the telegram send/ recieve sequence. The dependency of the telegram position of the IEC user program can be avoided.

Diagram 5. shows how data is sent from the EtherCAT master device to a slave device. The shown sequence is valid for synchronous operations and master setting “FrameAtStart=true”. For instance data n is calculated in cycle n (range C), it is sent via telegram at the beginning of the next cycle n+1 (range B, F) and arrives in the slave fieldbus hardware (range G). After the next synch interrupt (event s) the available data is exchanged with the slave firmware (range H) and computed in the slave specific control sequence (range I). The positon of the telegram arrival (range G) depends on the adjusted cycletime, adjusted synch offset, further time constants and jitter depending on hardware and used software stack.

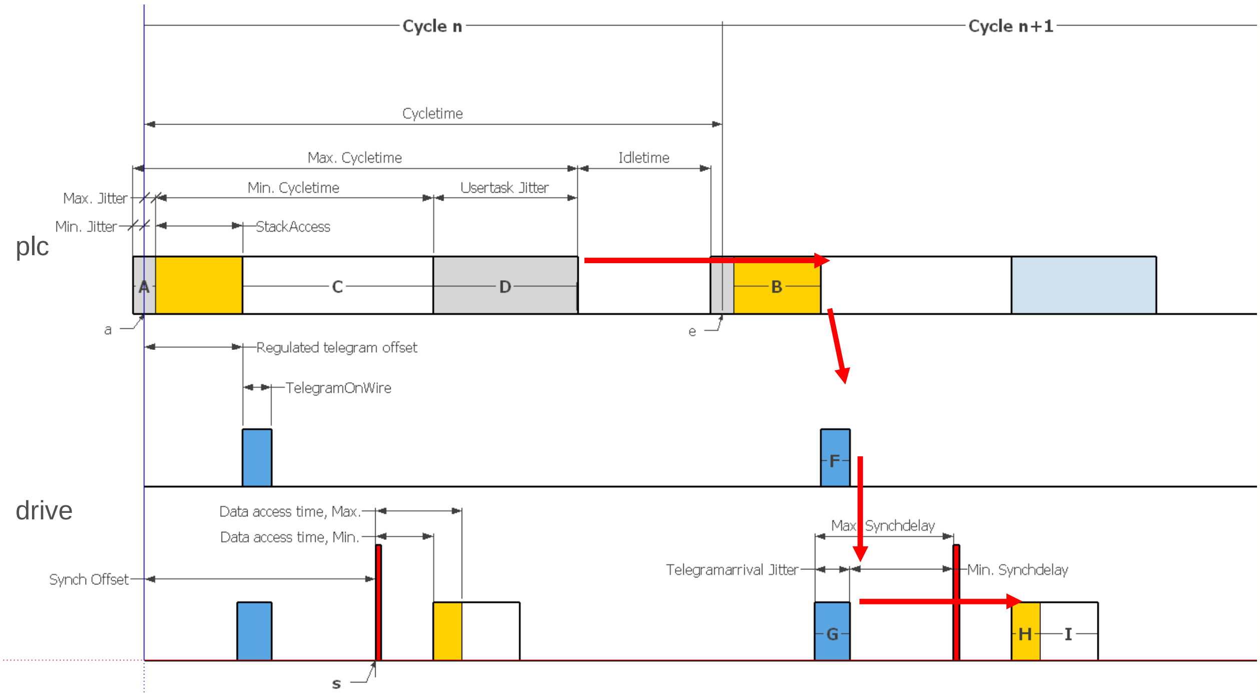


Diagram 5 - Send cycle, frame at start

a	Average cycle-start event, Execution start of plc cycle n
e	Average end-of-cycle n, execution start of plc cycle n+1
A	Jitter of PLC task startevent.
B	Stack access to read in data.
C	User code execution. Interpretation of current process values and calculation of set values to control a technical process.
D	Jitter of usercode execution time
E	Stack access to write out data and send the ethercat telegram.
Cycletime	Configured EtherCAT-Bus Cycle time (Setting of EtherCAT-Master / EtherCAT-Task).

Max. Cycletime: Maximum measured runtime of PLC cycle.
Min. Cycletime: Minimum measured runtime of PLC cycle.
Idletime: Buffer/ Idle time at the end of the PLC task until startevent of next cycle.
Max. Jitter Jitter of PLC task, maximum value.
Min. Jitter Jitter of PLC task, minimum value.
Stack Access Time needed for stack handling and to read process input data (including stack related offset time)

Calculation

The calculation shows, how to estimate the telegram arrival position **without** using the EtherCAT wizard.

This step can be skipped, if measured data from the EtherCAT wizard is available.

Sample:

SynchOffset:	20%:
Cycletime:	4ms
Data access time, Max (S6K):	150µs
Data access time, Min (S6K):	50µs
TelegramOnWire:	50µs
Min. Jitter:	-40µs
Max. Jitter:	+40µs
StackAccess C6C2	100µs
Shifttime	0µs (default)

*CycleStart = - SynchOffset *Cycletime*
*CycleStart := - 0.2*4000µs:= -800 µs(before synch):= 3200µs (after previous synch)*

Max. Synchdelay = CycleStart + StackAccess *+ Min. Jitter + TelegramOnWire - Shifttime*
Max. Synchdelay = -800 µs + 100µs + (-40) µs + 50 µs – 0 = -690µs

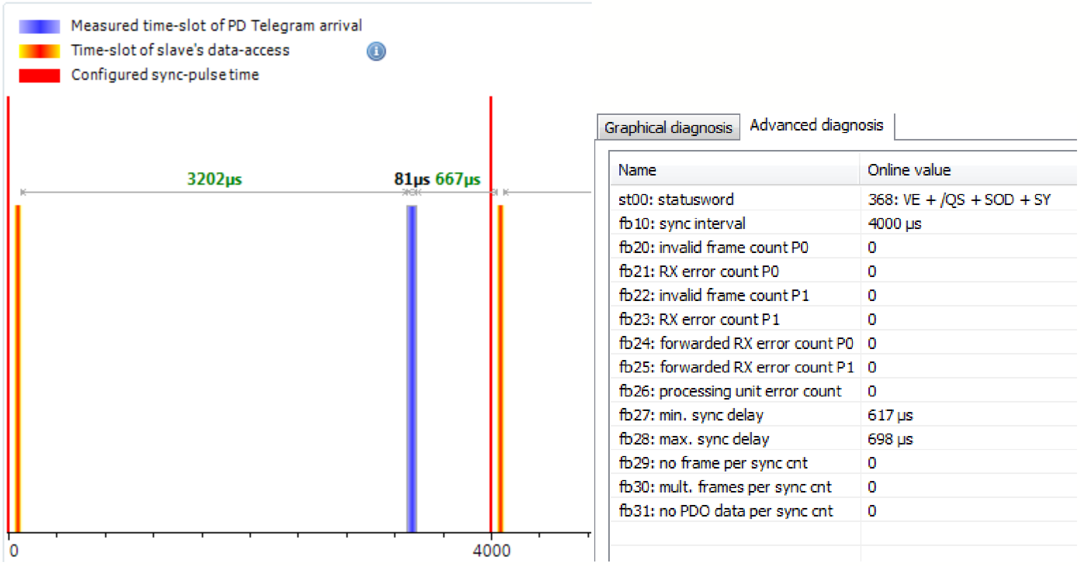
Min. Synchdelay = CycleStart + Max. Jitter + TelegramOnWire
Min. Synchdelay = -800 µs + 100 *µs + 40 µs + 50 µs - 0= -610 µs*

Telegramarrival Jitter= |Max. Synchdelay – Min. Synchdelay|= |-690µs – (-610µs)| = 80 µs

Tolerance time 1 (see diagram 4, ethercat wizard):= Cycletime - *Max. Synchdelay- Data access time, Max*
Tolerance time 1 = 4000µs – *(690µs) -150µs=* ***3210µs***

Tolerance time 2 (see diagram 4, ethercat wizard):= *Data access time, Min - Min. Synchdelay*
Tolerance time 2 = *610µs – (-50µs)=* ***660 µs***

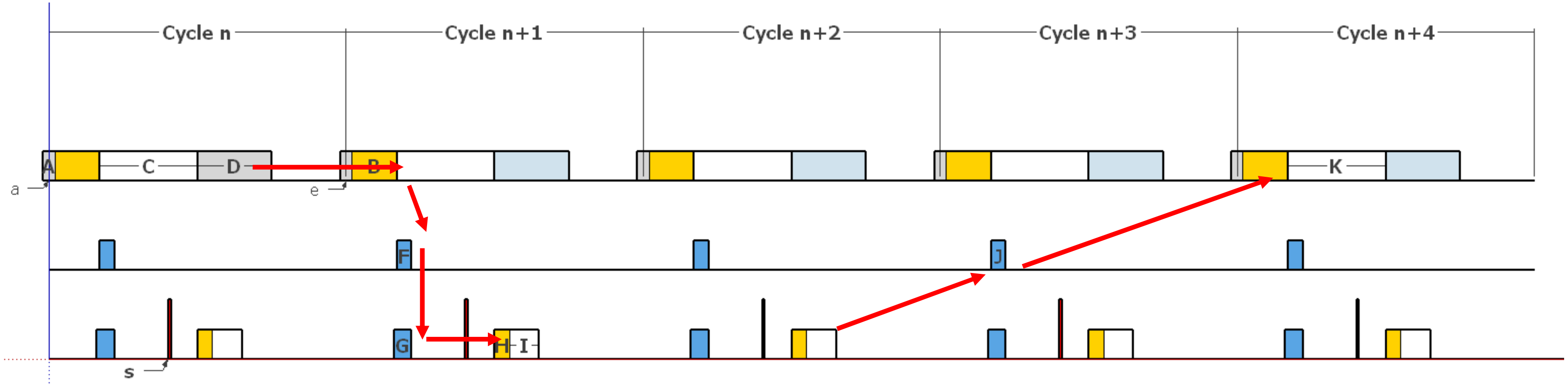
Measurement


Measured time-slot of PD Telegram arrival
Time-slot of slave's data-access
Configured sync-pulse time
3202µs
81µs
667µs
0
4000
Graphical diagnosis
Advanced diagnosis

Name	Online value
st00: statusword	368: VE + /QS + SOD + SY
fb10: sync interval	4000 µs
fb20: invalid frame count P0	0
fb21: RX error count P0	0
fb22: invalid frame count P1	0
fb23: RX error count P1	0
fb24: forwarded RX error count P0	0
fb25: forwarded RX error count P1	0
fb26: processing unit error count	0
fb27: min. sync delay	617 µs
fb28: max. sync delay	698 µs
fb29: no frame per sync cnt	0
fb30: mult. frames per sync cnt	0
fb31: no PDO data per sync cnt	0

Interpretation:
Telegram arrival is an save range and has acceptable jitter. As the arrival is not influenced by the user task runtime, this setting can be used for synchronous operations.

Send/ recieveCycle



C: PDO data n is calculated in PLC cycle
F: Data n on wire. The data arrives the slave with the falling edge of this range.
G: Telegram arrival range.
H: PDO buffer is read/ written by slave.
I: Data n is processed by slave firmware.
J: data n on wire in direction of PLC
K: Data n re-read in PLC

Summary: frame at task start=true

Advantage:

The telegram arrival is not effected by the duration of the user program. The stack execution start is controlled strictly, so a small jitter is expected.

If the hardware dependend offset is known, the position of the telegram arrival can be precalculated without measuring the user program runtime.

The user program runtime can be decreased/increased much more flexible without checking the telegram arrival again. Anyway, the user program runtime should be checked to avoid cycletime overflows.

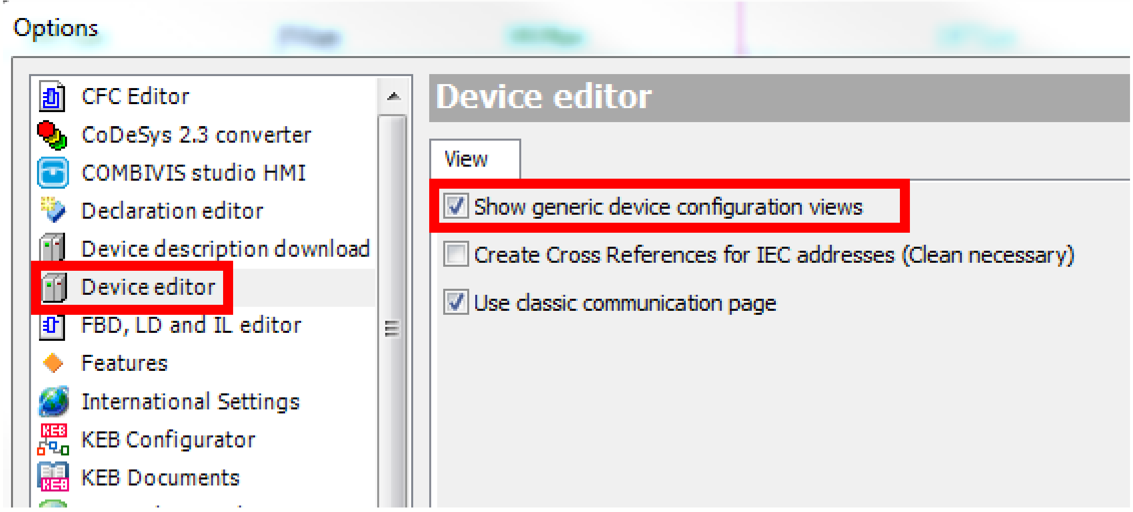
Disadvantage:

Pdo are sent and received one cycle delayed. A write/ read back loop takes 5 cycles.

How to change setting for frame send event?

First the generic device editor view has to be activated:

Tools→Options→Device Editor→Show generic device configuration views



Device editor

View

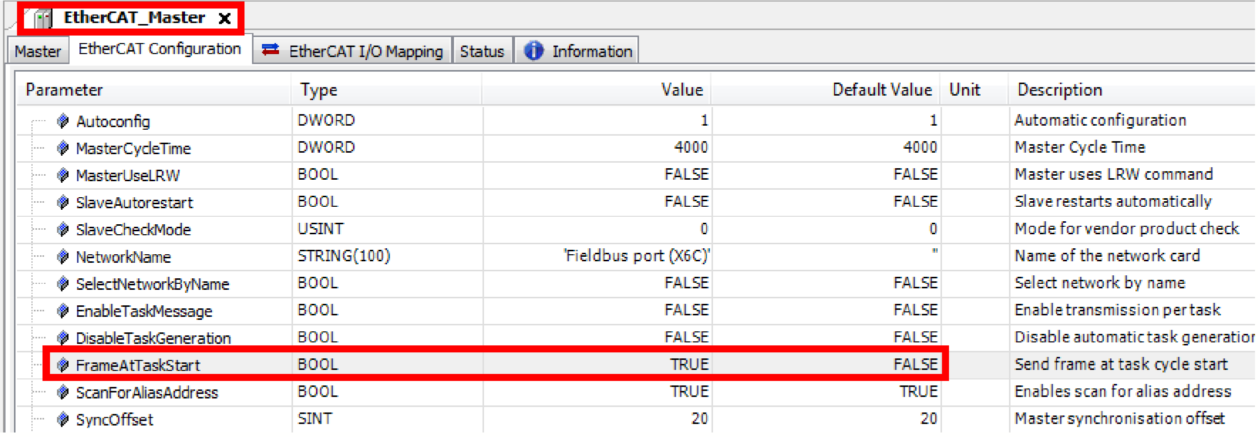
☑ Show generic device configuration views

□ Create Cross References for IEC addresses (Clean necessary)

☑ Use classic communication page

Now the property can be changed using the EtherCAT master device.

EtherCAT master→ EtherCAT Configuration → FrameAtTaskStart



Parameter	Type	Value	Default Value	Unit	Description
Autoconfig	DWORD	1	1		Automatic configuration
MasterCycleTime	DWORD	4000	4000		Master Cycle Time
MasterUseLRW	BOOL	FALSE	FALSE		Master uses LRW command
SlaveAutorestart	BOOL	FALSE	FALSE		Slave restarts automatically
SlaveCheckMode	USINT	0	0		Mode for vendor product check
NetworkName	STRING(100)	'Fieldbus port (X6C)'	''		Name of the network card
SelectNetworkByName	BOOL	FALSE	FALSE		Select network by name
EnableTaskMessage	BOOL	FALSE	FALSE		Enable transmission per task
DisableTaskGeneration	BOOL	FALSE	FALSE		Disable automatic task generatior
FrameAtTaskStart	BOOL	TRUE	FALSE		Send frame at task cycle start
ScanForAliasAddress	BOOL	TRUE	TRUE		Enables scan for alias address
SyncOffset	SINT	20	20		Master synchronisation offset

Annex

Table of KEB PLC timing

Name	Version	Stack	Stack Access time [µs]
C6C2	3.4.1.7	3.5.3.50	100
C6 SMART	3.5.6.60	3.5.6.40	100

Table of KEB DRIVE timing

Name	Version	Data access, Min [µs]	Data access, Max [µs]
H6	1210,1211,1220,1221,1230,1231,1240,1241,1250,1251	50	100
S6K	9241,9242	50	100

Disclaimer

KEB Automation KG reserves the right to change/adapt specifications and technical data without prior notification. The safety and warning reference specified in this manual is not exhaustive. Although the manual and the information contained in it is made with care, KEB does not accept responsibility for misprint or other errors or resulting damages. The marks and product names are trademarks or registered trademarks of the respective title owners.

The information contained in the technical documentation, as well as any user-specific advice in verbal or in written form are made to the best of our knowledge and information about the application. However, they are considered for information only without responsibility. This also applies to any violation of industrial property rights of a third-party.

Inspection of our units in view of their suitability for the intended use must be done generally by the user. Inspections are particular necessary, if changes are executed, which serve for the further development or adaption of our products to the applications (hardware, software or download lists). Inspections must be repeated completely, even if only parts of hardware, software or download lists are modified.

Application and use of our units in the target products is outside of our control and therefore lies exclusively in the area of responsibility of the user.

KEB Automation KG
Südstraße 38 • 32683 Barntrup, Germany
fon: +49 5263 401-0 • fax: +49 5263 401-116
net: www.keb.de • mail: info@keb.de