acontis technologies GmbH

SOFTWARE

EC-Master

EtherCAT® Master Stack Class A

Version 3.2

Edition: November 8, 2023

Contents

1 Synchronization with Distributed Clocks (DC)

DC clock synchronization enables all EtherCAT devices (master and slaves) to share the same EtherCAT System Time.

A “DC-slave” is defined as slave who shall be synchronized by means of distributed clocks. During network start-up several steps have to be performed by the EC-Master to set-up a consistent time base in all DC-slaves:

- Initial propagation delay measurement and compensation (ETG.8000)
- Offset compensation (ETG.8000)
- Set start time (ETG.8000)
- After network start-up: continuous drift compensation (ETG.8000)
- The Master must synchronize itself on the reference clock (ETG.1020) -> DCM

Reference:

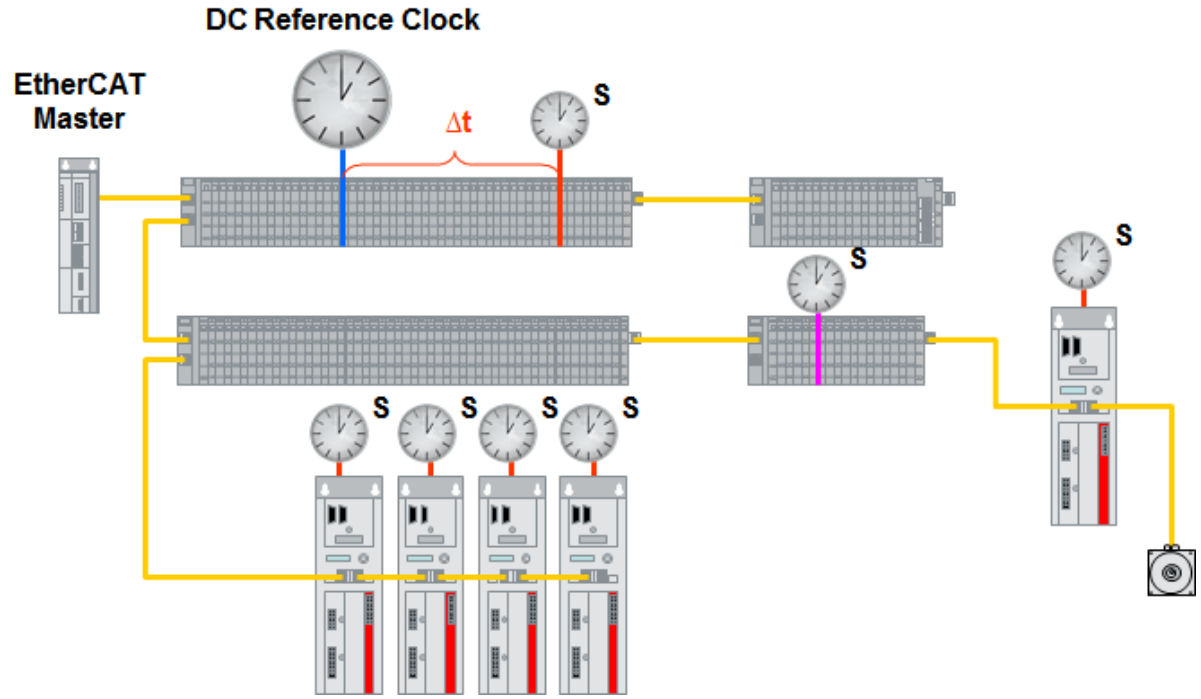
- ETG.1000.3 and ETG.1000.4
- ETG.1020 -> Synchronization
- ETG.8000 -> Distributed Clocks

1.1 Technical overview

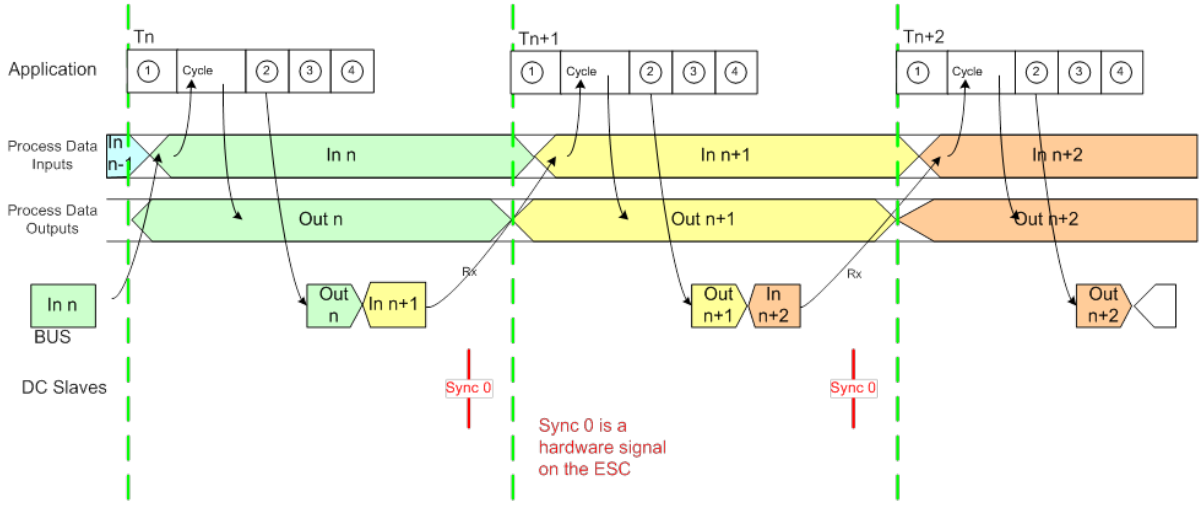
1.1.1 Support slaves and topologies

EC-Master supports all currently existing slave types and possible topologies:

- Slaves with 32 bit or 64 bit system time register (0x0910)
- Reference Clock with 32 bit or 64 bit system time register (0x0910)
- Reference Clock as first slave (auto increment address 0) or in the middle. Only slaves behind the reference clock could be synchronized.
- Drift compensation with 32 bit or 64 bit ARMW command in the cyclic frame
- Topologies: Line, Daisy chain, Daisy chain with drop lines, tree structure, star topology with EK1122 junctions



1.1.2 Typical Bus Timing



1.1.3 Implementation Details

The generation of Sync impulses is started after initialization and sending of 10000 FRMW frames which are required to bring slaves initially into sync. After this FRMW generation a grace period of 50 ms under real-time OSes is used and 500 ms using Windows is configured before start of cyclic operation of slaves, which causes the Sync impulse generation delayed by this grace period.

Initial propagation delay measurement and offset compensation commands are not part of the ENI file. The ARMW command for drift compensation is part of the cyclic frame.

1.1.4 Slaves in sync

Slaves in sync means that the system time difference of all DC slaves do not exceed a configured limit. Out of sync is detected individually and immediately for each slaves.

The master awaits that the slaves are in sync in Master state transition INIT->PREOP. Therefore the master state transition may timeout if the slaves do not get in sync.

Due to technology the slaves are always getting in sync as long as there is no error in setup. In order to detect system time difference exceeding, *Sync Window Monitoring* is used.

1.1.5 Sync Window Monitoring

Sync Window Monitoring must be explicitly enabled in configuration.

The system time difference exceeding detection in Sync Window Monitoring uses a deviation limit and a settle time and is issued continuously with configured commands (Ado 0x092C) in cyclic frames.

In sync is assumed if there is no violation of the system time difference limit (for all DC slaves!) detected within the settle time. The deviation limit (dwDevLimit) and settle time (dwSettleTime) can be configured using `emDcConfigure()`.

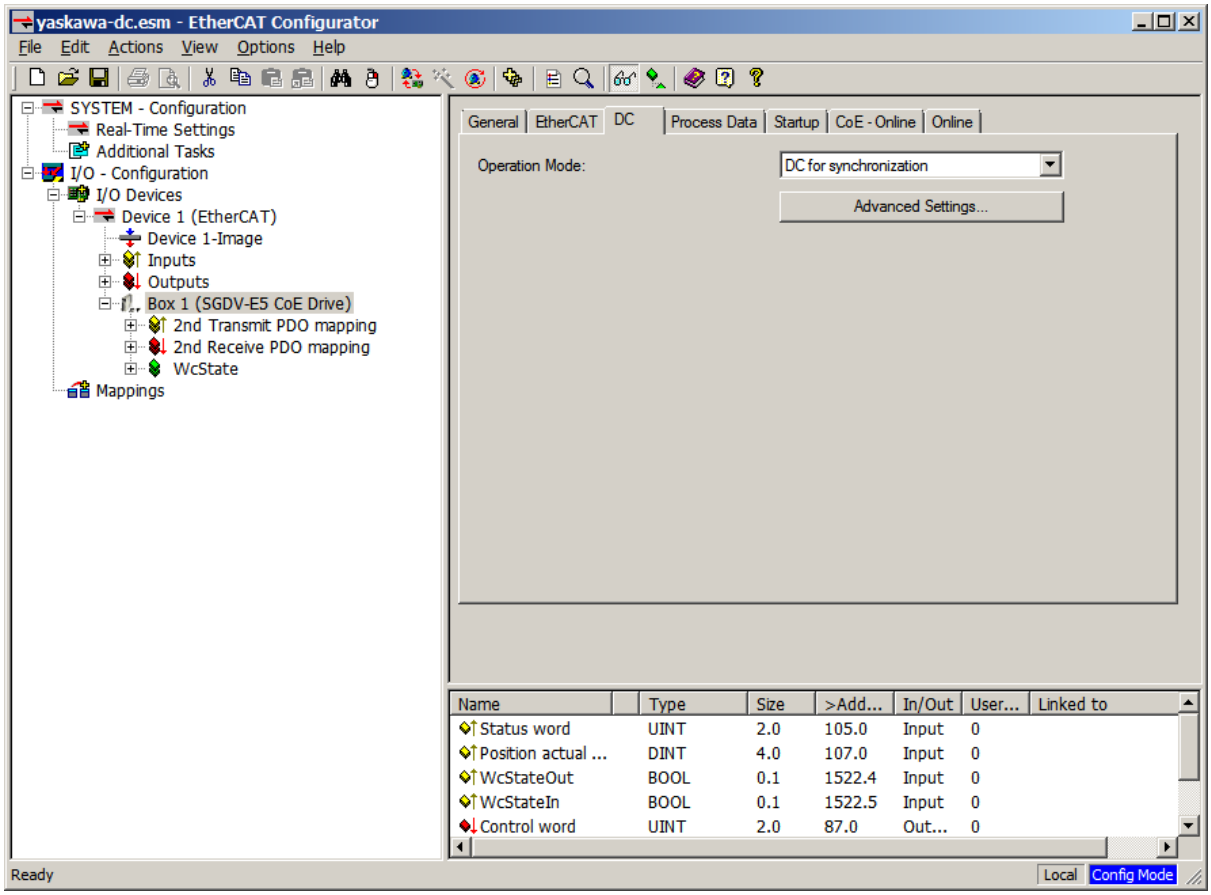
If the configuration only contains the cyclic commands for SAFE-OP or OP (e.g. ET9000) the master queues acyclic datagrams (Ado 0x092C) for system time difference measurement.

If there are at less than two DC slaves on bus (e.g. it the reference clock is the only DC slave on bus), Sync Window Monitoring is skipped. If it is skipped, because it is not enabled in configuration or there are less than two DC slaves on bus, slaves are immediately considered in sync.

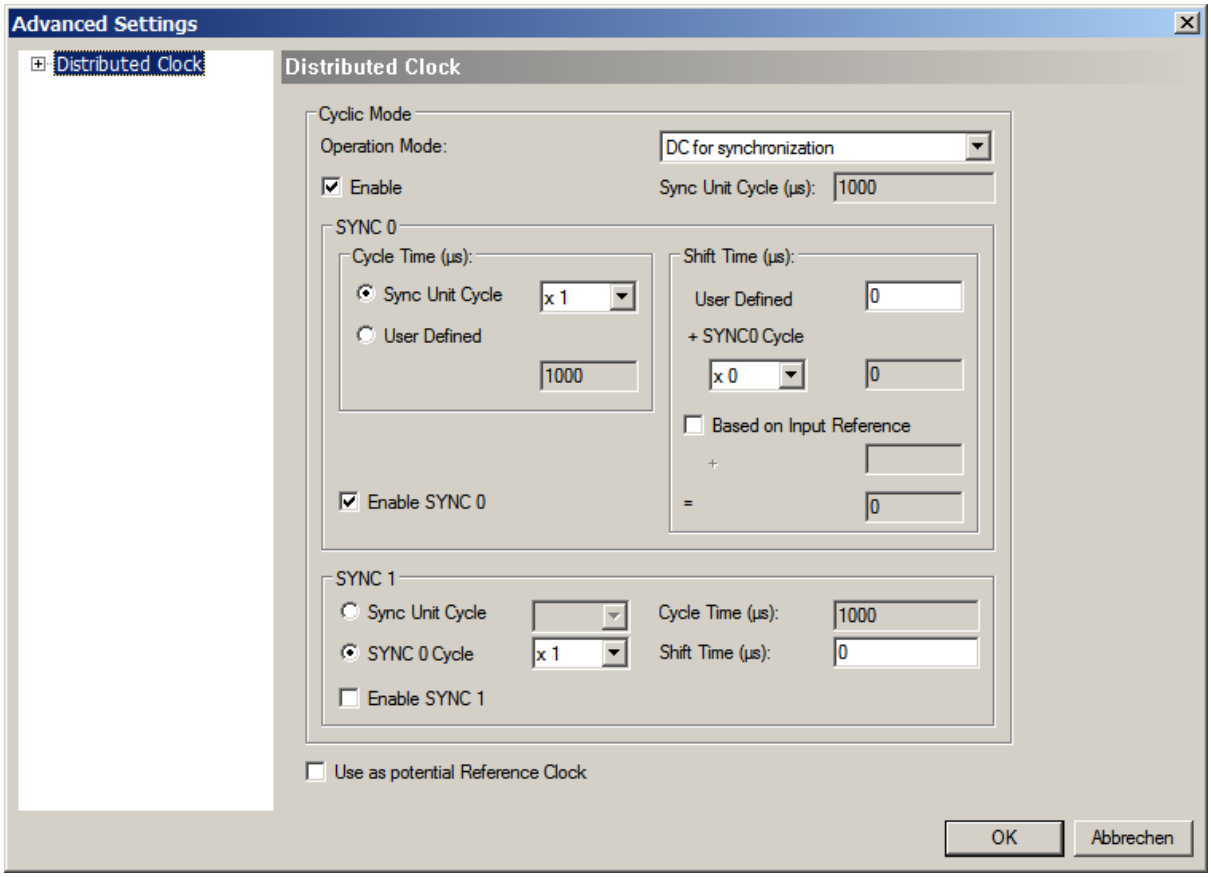
1.2 Configuration with ET-9000

1.2.1 Enable “DC Mode” for slave

If a slave supports DC, an additional tab in ET9000 appears.

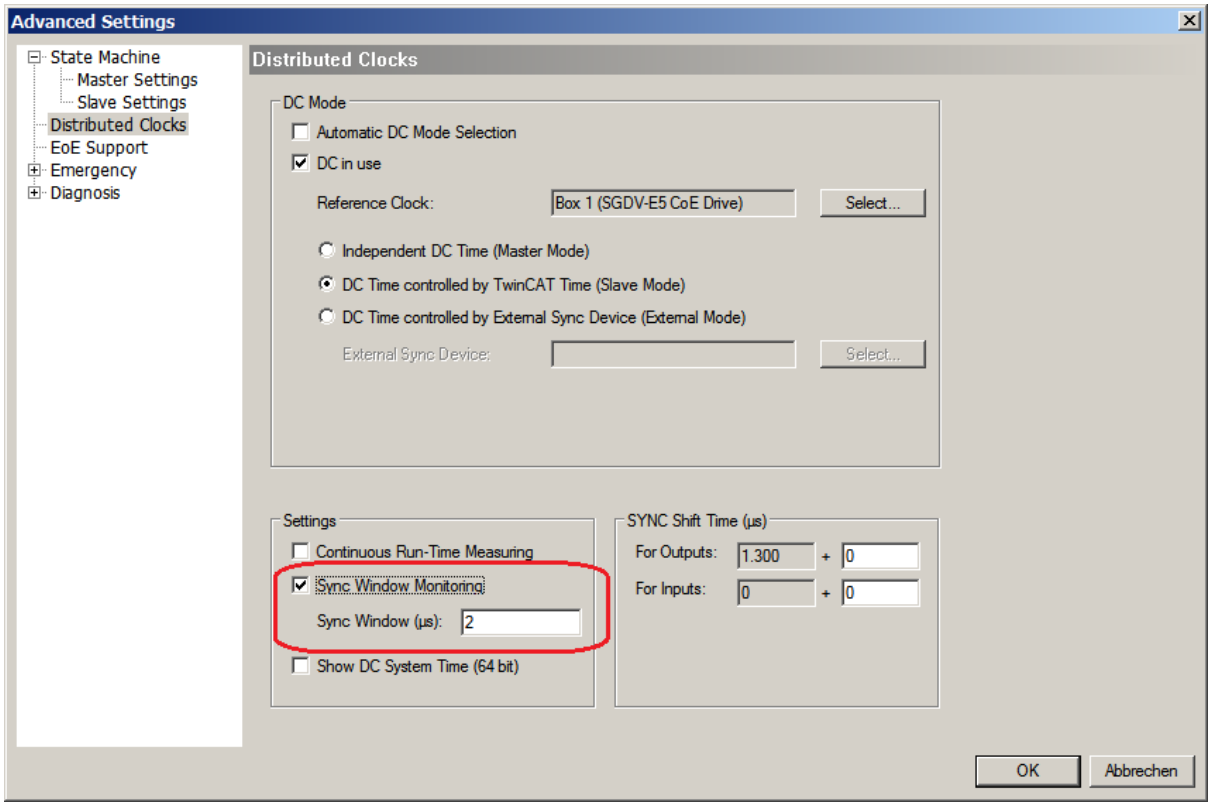


In the “Advanced Settings” additional slave specific parameters may be set. By default the cycle time for “SYNC 0” is equal to the bus cycle time (Sync Unit Cycle).



1.2.2 Enable Sync Window Monitoring for master

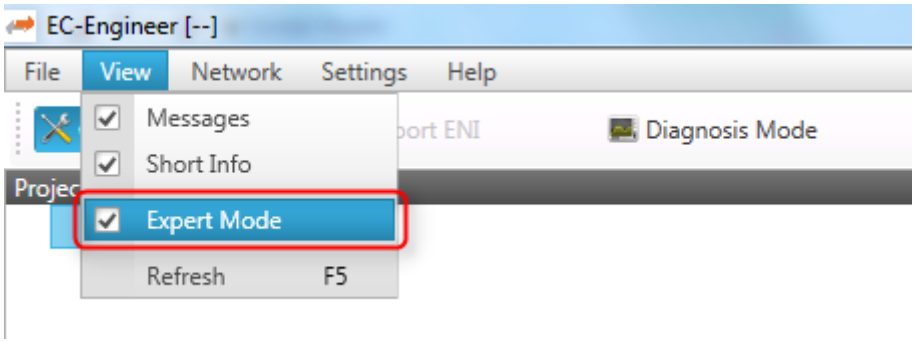
By enabling the option “Sync Window Monitoring” in the “Advanced Settings” of the master, the EtherCAT configurator will insert a command (datagram) in the cyclic frame to read the ESC registers 0x092C. If this is selected the master will throw the notification *emNotify - EC_NOTIFY_DC_SLV_SYNC*.



1.3 Configuration with EC-Engineer

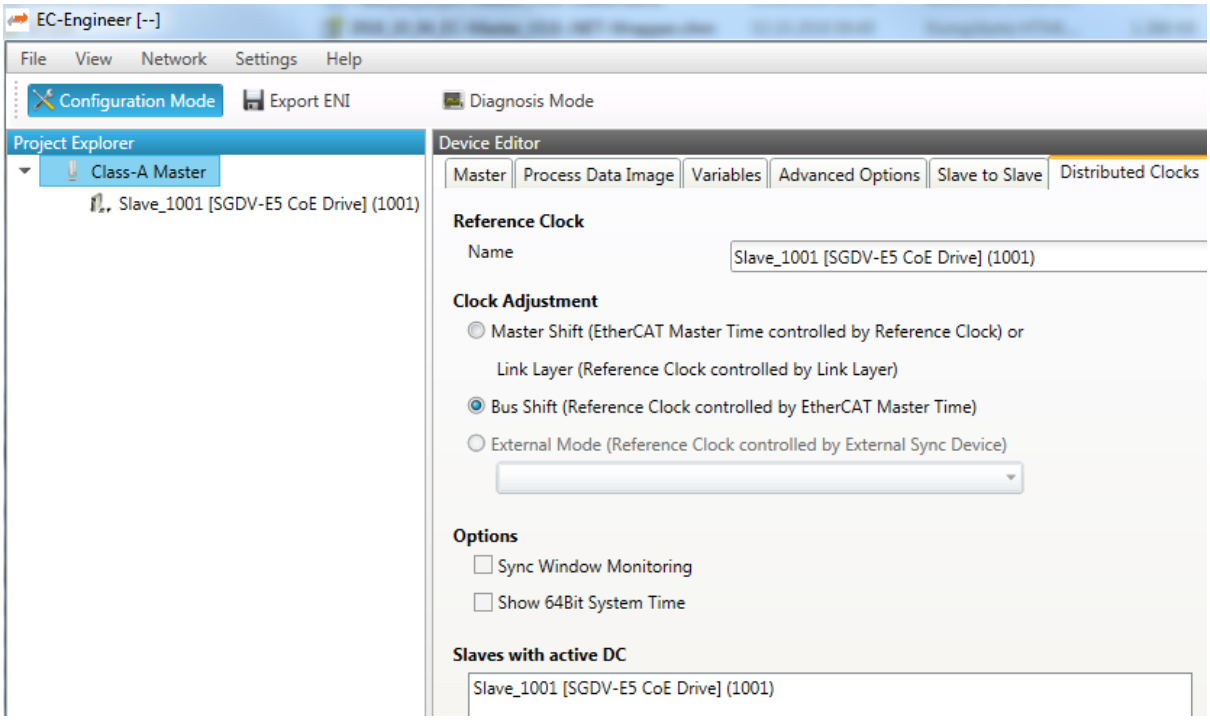
The EC-Engineer automatically choses the DC settings for slaves as proposed by the device's vendor and sets DCM mode to bus shift.

The settings can be changed according to the project's needs. DC options are part of the EC-Engineer Expert Mode. The Expert Mode can be activated from the menu:



1.3.1 Distributed Clocks Master settings (Expert)

In this tab, the user can change distributed clocks related settings. The tab is only available if the configuration contains slaves.

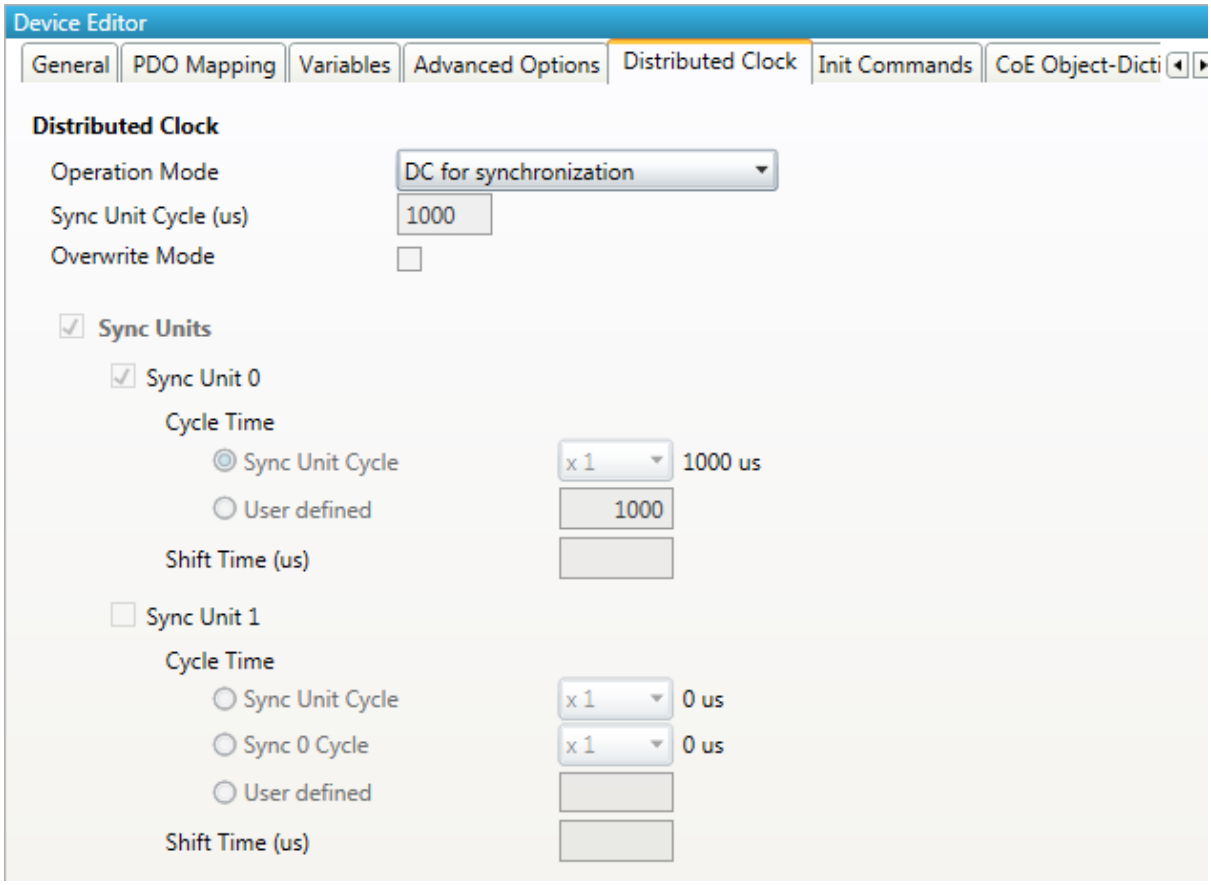


- Reference Clock Name: Name of the reference clock. By default, this is the first slave with DC support.
- Master Shift: The reference clock controls the Master time
- Bus Shift: The Master time controls the reference clock. A command will be inserted in the Cyclic frame to adjust the reference clock system time (write to register 0x0910).
- Continuous Propagation Compensation: A command will be inserted in the Cyclic frame which allows the EtherCAT master to measure and compensate the propagation delay time by time.
- Sync Window Monitoring: A command will be inserted in the cyclic frame to read the ESC registers 0x092C. If this is selected the master will throw a notification.
- Show 64Bit System Time: Master supports slaves with 32bit and 64bit system time register (0x0910). If this is selected he will interpret it as 64bit system time.

Note: If no reference clock is displayed, please ensure at least one slave in the network is configured for DC operation mode *Distributed Clocks Slave settings (Expert)*

1.3.2 Distributed Clocks Slave settings (Expert)

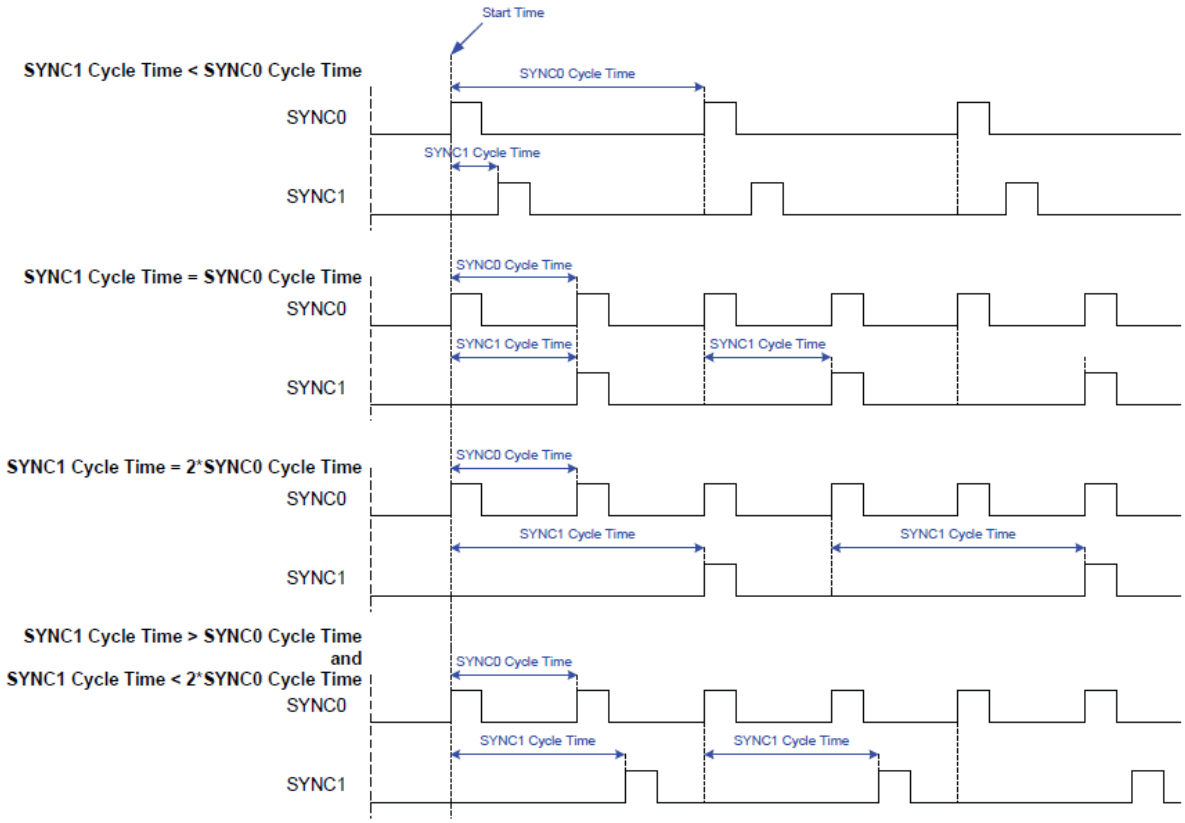
In this tab, the user can change distributed clocks related settings. The tab is only available if the device's vendor specified the DC usage. Sync signal generation or DC latching is selected automatically according to Operation Mode.



- Operation Mode: Selectable DC operation modes. The modes cannot be edited.
- Sync Unit Cycle: Base interval in microseconds which will be used from master. The Sync Units can be activated and configured to generate signals.

See also:

SyncSignal Generation in the ET1100 Datasheet for time describing



1.4 Programmer's Guide

1.4.1 emDcConfigure

static EC_T_DWORD **ecatDcConfigure** (struct _EC_T_DC_CONFIGURE *pDcConfigure)

EC_T_DWORD **emDcConfigure** (
    EC_T_DWORD dwInstanceID,
    EC_T_DC_CONFIGURE *pDcConfigure
)

Configure the distributed clocks.

- Set the DC synchronization settling time ([ms]).
- Set the DC slave limit for the wire or'ed clock deviation value. This value determines whether the slave clocks are synchronized or not.
- Configure the ARMW burst frames to compensate the static deviations of the clock speeds.

Parameters

- **dwInstanceID** – [in] Instance ID (Multiple EtherCAT Network Support)
- **pDcConfigure** – [in] Configuration parameter a pointer to a structure of type *EC_T_DC_CONFIGURE*.

Returns

EC_E_NOERROR or error code

struct **EC_T_DC_CONFIGURE**

Public Members

EC_T_DWORD **dwClntId**
[in] Reserved

EC_T_DWORD **dwTimeout**
[in] Timeout [ms] for the DC initialization in which time offsets and propagation delays are evaluated.

EC_T_DWORD **dwDevLimit**

[in] Maximum permissible deviation of the individual slave clock and the DC reference clock. The maximum deviation is determined by wire or'ed the deviations of the individual slave clocks with one another. The check against the limit is only active if "Sync Window Monitoring" is set in the configuration tool (EC Engineer), which generates a BRD command to read the slave register 0x092C in every cycle. The limit is calculated as follows:

2^n - 1 ns, e.g. a dwDevLimit of 4 corresponds to 14 ns.

A value of 0 disables the "Sync Window Monitoring"

EC_T_DWORD **dwSettleTime**
[in] Settle time [ms]. At the beginning of the synchronization the slave clocks oscillate strongly. To prevent multiple in-sync and out-of-sync notifications from being generated, a settling time can be set in which no notifications are generated.

EC_T_DWORD **dwTotalBurstLength**
[in] Overall amount of burst frames sent. Default 10000.

EC_T_DWORD **dwBurstBulk**
[in] Amount of burst frames per cycle during initialization burst. Default 12.

EC_T_BOOL **bBulkInLinkLayer**
[in] If EC_TRUE, bulk is realized by link layer, otherwise by master. The MAC needs to support the frame repeating function. In this case the link layer will repeat the DC burst frames itself, reducing the hardware accesses of the master to the MAC.

EC_T_BOOL **bAcycDistributionDisabled**
[in] If EC_TRUE, acyclic distribution is disabled

EC_T_DWORD **dwDcStartTimeGrid**
[in] Time grid [ns] to align DC start time. With the help of the grid, several EtherCAT networks can be synchronized without a random shift value between the SYNC signals.

EC_T_BOOL **bDcInitBeforeSlaveStateChange**
[in] If EC_TRUE, DC is initialized before slaves state change to PREOP

EC_T_DWORD **dwReserved**[4]
[in/out] Reserved

See also:

Chapter "Drift Compensation" of the ETG Document "ESC Datasheet Section 1 - Technology"

1.4.2 emDcIsEnabled

static EC_T_DWORD **ecatDcIsEnabled**(EC_T_BOOL *pbDcIsEnabled)

EC_T_DWORD **emDcIsEnabled**(EC_T_DWORD dwInstanceID, EC_T_BOOL *pbDcIsEnabled)
Determines if DC is enabled and used.

Parameters

- **dwInstanceID** – [in] Instance ID (Multiple EtherCAT Network Support)
- **pbDcIsEnabled** – [out] EC_TRUE if DC is enabled

Returns
EC_E_NOERROR or error code

1.4.3 emGetBusTime

static EC_T_DWORD **ecatGetBusTime**(EC_T_UINT64 *pqwBusTime)

EC_T_DWORD **emGetBusTime**(EC_T_DWORD dwInstanceID, EC_T_UINT64 *pqwBusTime)
This function returns the actual bus time in nanoseconds.

Parameters

- **dwInstanceID** – [in] Instance ID (Multiple EtherCAT Network Support)
- **pqwBusTime** – [out] Bus time [ns]

Returns
EC_E_NOERROR or error code

1.4.4 emDcContDelayCompEnable

static EC_T_DWORD **ecatDcContDelayCompEnable**(EC_T_VOID)

EC_T_DWORD **emDcContDelayCompEnable**(EC_T_DWORD dwInstanceID)
Enable the continuous propagation delay compensation.

Calling this function generate a propagation delay measurement every 30s. The result of the measurement is used to correct the propagation delay values on the bus.

Parameters
dwInstanceID – [in] Instance ID (Multiple EtherCAT Network Support)

Returns
EC_E_NOERROR or error code

1.4.5 emDcContDelayCompDisable

static EC_T_DWORD **ecatDcContDelayCompDisable**(EC_T_VOID)

EC_T_DWORD **emDcContDelayCompDisable**(EC_T_DWORD dwInstanceID)
Disable the continuous propagation delay compensation.

Parameters
dwInstanceID – [in] Instance ID (Multiple EtherCAT Network Support)

Returns
EC_E_NOERROR or error code

1.4.6 emIoControl - EC_IOCTL_DC_SLV_SYNC_STATUS_GET

Get the last generated *emNotify - EC_NOTIFY_DC_SLV_SYNC* notification.

emIoControl - EC_IOCTL_DC_SLV_SYNC_STATUS_GET

Parameter

- `pbyInBuf`: [in] Should be set to EC_NULL
- `dwInBufSize`: [in] Should be set to 0
- `pbyOutBuf`: [out] Pointer to EC_T_DC_SYNC_NTFY_DESC data type
- `dwOutBufSize`: [in] Size of the output buffer in bytes
- `pdwNumOutData`: [out] Pointer to EC_T_DWORD. Amount of bytes written to the output buffer pbyOutBuf

Return
EC_E_NOERROR or error code

See also:

emNotify - EC_NOTIFY_DC_SLV_SYNC describes *EC_T_DC_SYNC_NTFY_DESC*

1.4.7 emIoControl - EC_IOCTL_DC_SETSYNCSTARTOFFSET

Set the safety offset applied to the “set DC start time” InitCmd during the PS transition.

emIoControl - EC_IOCTL_DC_SETSYNCSTARTOFFSET

Parameter

- `pbyInBuf`: [in] Pointer to EC_T_DC_STARTCYCSAFETY_DESC data type
- `dwInBufSize`: [in] Size of the input buffer provided at pbyInBuf in bytes.
- `pbyOutBuf`: [out] Should be set to EC_NULL
- `dwOutBufSize`: [in] Should be set to 0
- `pdwNumOutData`: [out] Should be set to EC_NULL

Return
EC_E_NOERROR or error code

struct **EC_T_DC_STARTCYCSAFETY_DESC**

Public Members

EC_T_DWORD **dwStartCycSafetyLo**
[in] Start SYNC Cyc Safety [ns] Lower 32 Bit

EC_T_DWORD **dwStartCycSafetyHi**
[in] Start SYNC Cyc Safety [ns] Upper 32 Bit

Default: 50000000ns

1.4.8 emIoControl - EC_IOCTL_DC_FIRST_DC_SLV_AS_REF_CLOCK

Enable or disable the usage of the first DC slave on bus overriding the configured reference clock.

emIoControl - EC_IOCTL_DC_FIRST_DC_SLV_AS_REF_CLOCK

Parameter

- `pbyInBuf`: [in] pointer to EC_T_BOOL. EC_FALSE: disable, EC_TRUE: enable.
- `dwInBufSize`: [in] Size of the input buffer provided at pbyInBuf in bytes.
- `pbyOutBuf`: [out] Should be set to EC_NULL
- `dwOutBufSize`: [in] Should be set to 0
- `pdwNumOutData`: [out] Should be set to EC_NULL

Return
EC_E_NOERROR or error code

1.4.9 emFindInpVarByName - "Inputs.BusTime"

The DC system time (written to ESC register 0x0910) is part of the process data with name "Inputs.BusTime".

static EC_T_DWORD **ecatFindInpVarByName** (
    const EC_T_CHAR *szVariableName,
    EC_T_PROCESS_VAR_INFO *pProcessVarInfoEntry
)
EC_T_DWORD **emFindInpVarByName** (
    EC_T_DWORD dwInstanceID,
    const EC_T_CHAR *szVariableName,
    EC_T_PROCESS_VAR_INFO *pProcessVarInfoEntry
)

See also:

`emFindInpVarByName()` in the EC-Master Class B documentation

1.4.10 emIoControl - EC_IOCTL_DC_ENABLE_ALL_DC_SLV

Enable or disable the usage of DC at all supporting slaves on bus overriding the configured settings. Perhaps *emIoControl - EC_IOCTL_DC_FIRST_DC_SLV_AS_REF_CLOCK* is necessary to set the reference clock at an allowed position.

emIoControl - EC_IOCTL_DC_ENABLE_ALL_DC_SLV

Parameter

- `pbyInBuf`: [in] pointer to EC_T_BOOL. EC_FALSE: disable, EC_TRUE: enable.
- `dwInBufSize`: [in] Size of the input buffer provided at pbyInBuf in bytes.
- `pbyOutBuf`: [out] Should be set to EC_NULL
- `dwOutBufSize`: [in] Should be set to 0
- `pdwNumOutData`: [out] Should be set to EC_NULL

Return
EC_E_NOERROR or error code

1.4.11 emNotify - EC_NOTIFY_REFCLOCK_PRESENCE

Distributed clocks reference clock presence notification. It will be received before *emNotify - EC_NOTIFY_DC_SLV_SYNC* as soon as reference clock was found on bus or removed from bus.

This notification is disabled by default.

emNotify - EC_NOTIFY_REFCLOCK_PRESENCE

Parameter

- `pbyInBuf`: [in] pointer to notification descriptor EC_T_REFCLOCK_PRESENCE_NTFY_DESC
- `dwInBufSize`: [in] sizeof(EC_T_REFCLOCK_PRESENCE_NTFY_DESC)
- `pbyOutBuf`: [out] Should be set to EC_NULL
- `dwOutBufSize`: [in] Should be set to 0
- `pdwNumOutData`: [out] Should be set to EC_NULL

struct **`EC_T_REFCLOCK_PRESENCE_NTFY_DESC`**

Public Members

EC_T_BOOL **`bPresent`**
[in] Reference clock present

EC_T_SLAVE_PROP **`SlaveProp`**
[in] Slave properties

See also:

emIoControl - EC_IOCTL_SET_NOTIFICATION_ENABLED in the EC-Master Class B documentation for how to control the activation

1.4.12 emNotify - EC_NOTIFY_DC_STATUS

Distributed clocks status notification. It will be received after *emNotify - EC_NOTIFY_DC_SLV_SYNC* as soon as DC is initialized or topology change was done . After topology was changed it may be received without *emNotify - EC_NOTIFY_DC_SLV_SYNC* if slaves did not get out of sync.

If EC_E_NOERROR is returned and window monitoring is enabled, all slaves are in SYNC

emNotify - EC_NOTIFY_DC_STATUS

Parameter

- `pbyInBuf`: [in] Pointer to EC_T_DWORD (EC_E_NOERROR on success, Error code otherwise)
- `dwInBufSize`: [in] sizeof(EC_T_DWORD)
- `pbyOutBuf`: [out] Should be set to EC_NULL
- `dwOutBufSize`: [in] Should be set to 0
- `pdwNumOutData`: [out] Should be set to EC_NULL

See also:

emIoControl - EC_IOCTL_SET_NOTIFICATION_ENABLED in the EC-Master Class B documentation for how to control the deactivation

1.4.13 emNotify - EC_NOTIFY_DC_SLV_SYNC

DC slave synchronization notification. Every time the slaves are coming in sync or getting out of sync the clients will be notified here. The notification is raised in any case if any DC slaves are configured. Slaves can only be out of sync if *Sync Window Monitoring* is enabled otherwise they are considered in sync

This notification is enabled by default.

emNotify - EC_NOTIFY_DC_SLV_SYNC

Parameter

- `pbyInBuf`: [in] pointer to notification descriptor EC_T_DC_SYNC_NTFY_DESC
- `dwInBufSize`: [in] sizeof(EC_T_DC_SYNC_NTFY_DESC)
- `pbyOutBuf`: [out] Should be set to EC_NULL
- `dwOutBufSize`: [in] Should be set to 0
- `pdwNumOutData`: [out] Should be set to EC_NULL

struct **EC_T_DC_SYNC_NTFY_DESC**

Public Members

EC_T_DWORD **IsInSync**
: [in] EC_TRUE : Wire or'ed deviation value meets limit requirements. EC_FALSE: Wire or'ed deviation value does not meet limit requirements.The limit is set by ecatDcConfigure()

EC_T_DWORD **IsNegative**
: [in] EC_TRUE : deviation value is negative EC_FALSE: deviation value is positive

EC_T_DWORD **dwDeviation**
: [in] Wire or'ed deviation value [ns] in case of in sync

EC_T_SLAVE_PROP **SlaveProp**
: [in] Slave properties in case of out of sync

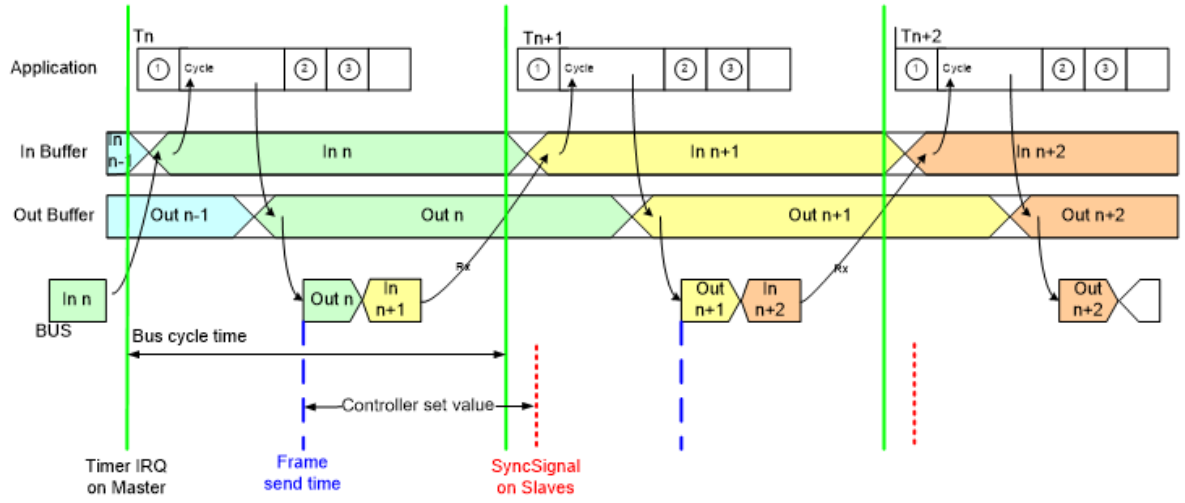
See also:

- *emDcConfigure()*
- emIoControl - EC_IOCTL_SET_NOTIFICATION_ENABLED in the EC-Master Class B documentation for how to control the deactivation

2 Master synchronization (DCM)

In applications like motion control it is necessary that process data update and the slave SYNC pulses are correlated in timely behavior, because otherwise the SYNC Interrupt (on slave) used to apply new process data to the application would use new data on some slaves and old data on some other in case the current cyclic datagram (which updates process data) is on the flow during the SYNC Interrupt is raised on all slaves at the same time.

The Distributed Clocks Master Synchronization (DCM) provides a controller mechanism to synchronize the process data update and the SYNC pulse in slaves.



The DC Master synchronization (DCM) in BusShift mode adjusts the bus time register of the DC reference clock. All the DC slaves converge to this time. This mode is useful to synchronize multiple EtherCAT busses or if adjustment of Master timer is not possible.

Features:

- PI drift controller
- Automatic timer adjustment error determination (I controller)

2.1 Technical overview

2.1.1 DCM Modes

The following DCM Modes are available:

Name	Purpose
BusShift	Synchronize slaves to master timer (Default)
MasterShift	Synchronize master timer to slave. Feasibility depending on target HW and SW.
LinkLayer-RefClock	Bus Shift using Link Layer clock. Special HW needed.
MasterRef-Clock	Bus Shift excluding reference clock controlling. Lowers CPU usages, but very high timer accuracy needed.
DCX	Synchronization of two or more EtherCAT segments by a a bridge device. Only available with Feature Pack External Synchronization.

See also:

emDcmConfigure()

2.1.2 Sync signal activation

The sync signals are activated during transition PREOP - SAFEOP according to Init Command (Ado 0x0980, 0x0990, 0x09A0, 0x09A8).

2.1.3 DCM in sync

DCM in sync means that the synchronization between the send time of the cyclic frames and the system time of the reference clock was successful.

The master awaits that DCM is in sync in Master state transition PREOP->SAFEOP. Therefore the master state transition may timeout if DCM does not get in sync. Due to the Master's DC implementation, DCM may get in sync in transition INIT->PREOP.

In sync is assumed if there is no error reported from the DCM controller within the settle time or if there is no DC slave connected.

2.1.4 Controller adjustment

To adjust the controller parameters the diagnostic values in file dcmlog0.0.csv can be used. The generation of logging information can be enabled setting bLogEnabled to `EC_TRUE` with the function *emDcmConfigure()*

Controller log file description:

Column name	Description
Time[ms]	Controller execution timestamp
SyncSetVal [ns]	Controller set value (distance between frame send time and SYNC0)
BusTime [ns]	System Time
BusTimeOff	System Time modulo Sync Cycle Time
CtlAdj [ns]	Controller adjust value
CtlErr [ns]	Controller error (EC_T_DCM_SYNC_NTFY_DESC.nCtlErrorNsecCur)
CtlErrFilt	Filtered controller error
Drift [ppm]	Drift between local clock and DC reference clock
CtlErr [1/10 pmil]	For internal use only
CtlOutSum	For internal use only
CtlOutTot	For internal use only
DCStartTime	DC start time send to the slaves to activate their SYNC signals
DCMErrorCode	Current error code of the DCM controller (same value as returned by emDcmGetStatus)
DCMInSync	Current InSync state of the DCM controller
DCInSync	Current InSync state of the DC slaves
SystemTimeDifference [ns]	Current system time difference of the DC slave if monitoring is enabled

Log file analyze: To understand how the controller values correlates the following table can helps.

Controller error	Drift	Reference-Clock	Output
Positive	Must decrease	Must run faster	Double cycle time
Negative	Must increase	Must run slower	Half cycle time

The DCM Controller reacts if the controller error is positive or negative. E.g. on a positive controller error (CtlErr) the drift is too high and has to be decreased. Therefore the controller will speed up the reference clock.

In some case the drift is too high (EcMasterDemoDc shows error messages) and cannot be balanced by the controller. This holds if the drift is higher than about 400ppm.

Diagram 1: Bus offset in nanoseconds

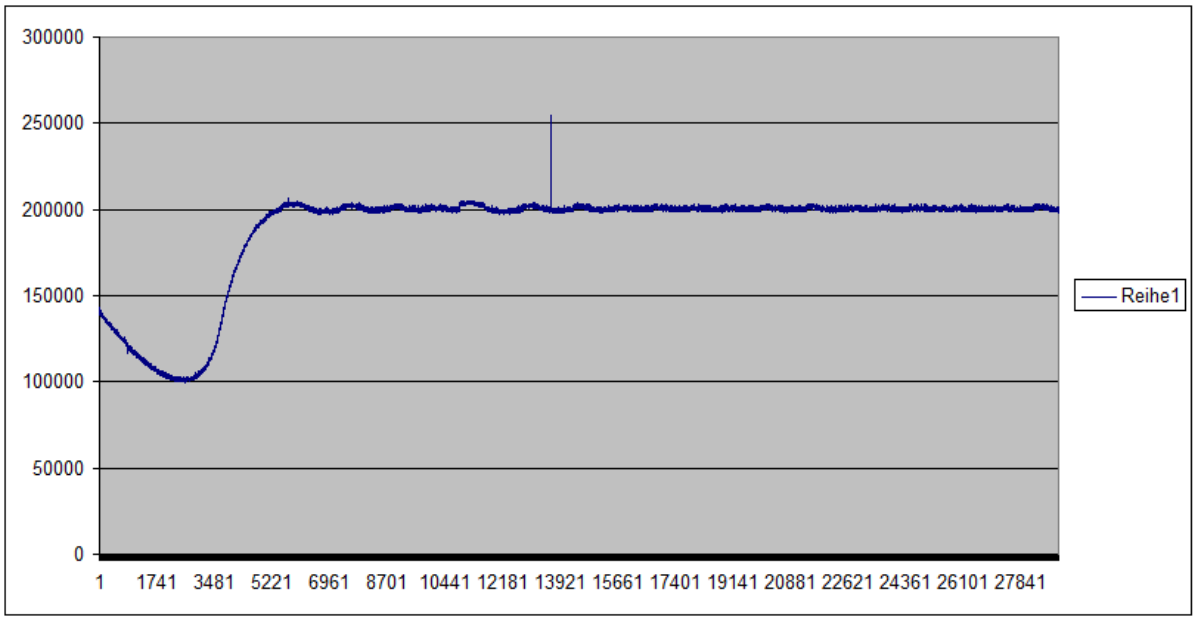


Diagram 2: Drift in ppm (part per million)

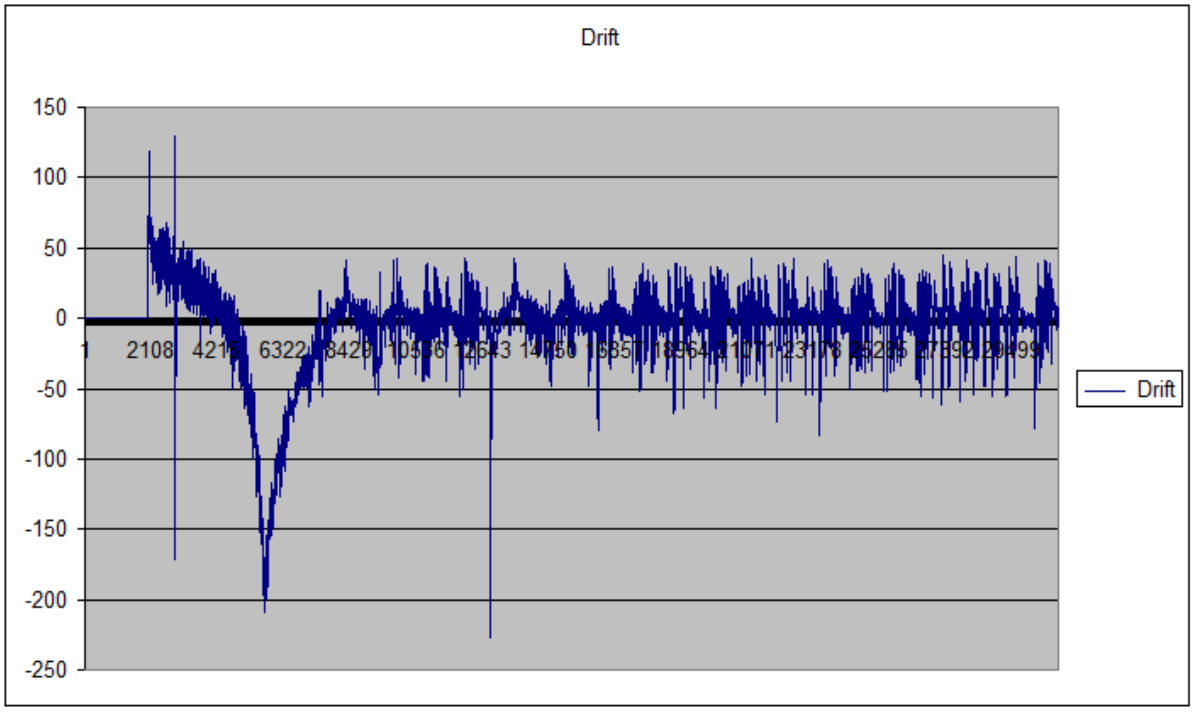


Diagram 3: Controller error in nanoseconds

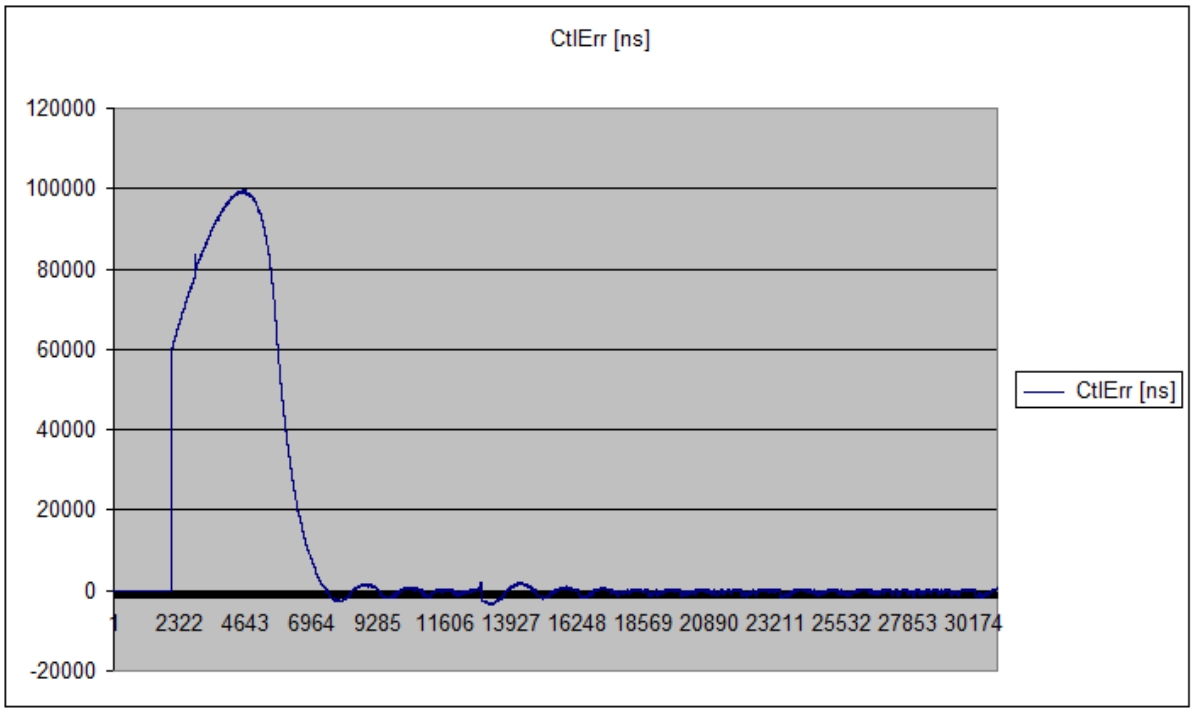
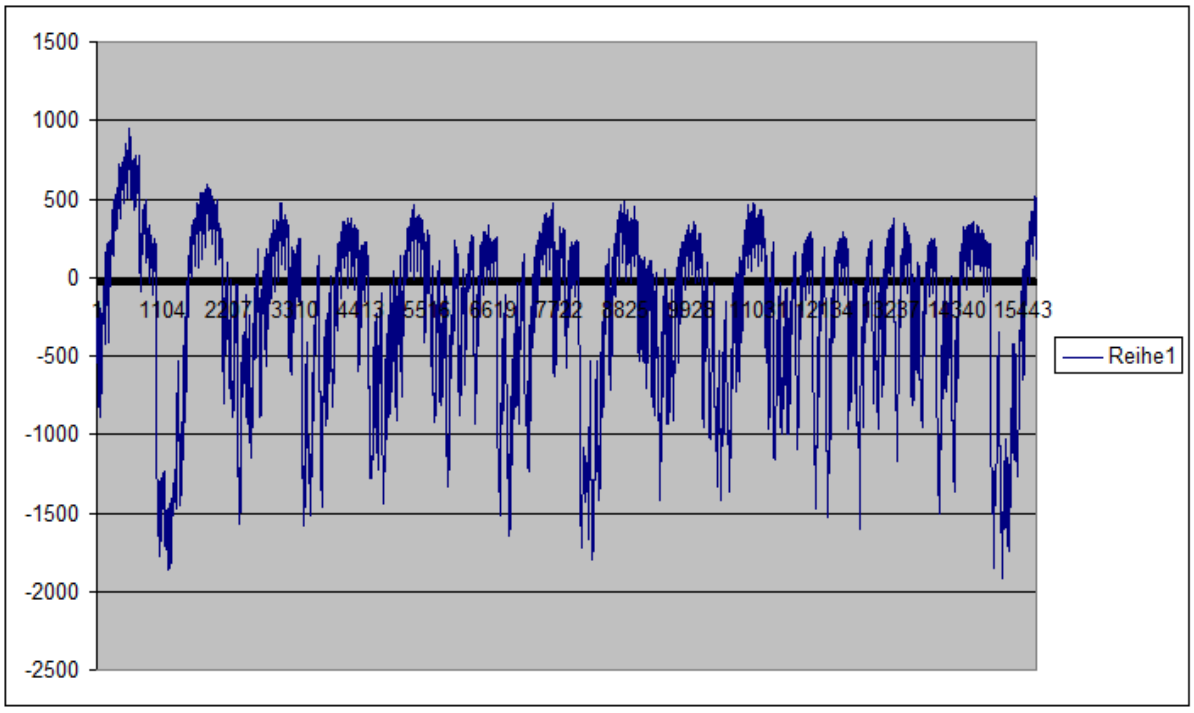


Diagram 4: Controller error in steady state in nanoseconds



Troubleshooting:

DCM BusShift needs a very deterministic and accurate time base.

The following statements have to be true:

- The timer input frequency must be determined with an accuracy greater than 600 ppm (333333 Hz vs 333000 Hz. E.g. at 1 ms, the cycle time must be between 999.400 µs and 1000.600 µs)
- The timer frequency must never change after the application start

On a PC platform the following settings have to be disabled in the BIOS. Be sure that these settings are really applied.

- System management interrupt
- Legacy USB support
- Intel C-STATE tech
- Intel Speedstep tech
- Spread Spectrum

2.1.5 DCM Master Shift mode

In this mode, the local time base will be adjusted to synchronize it with the network "bus time". The following function pointers have to be implemented to enable the adjustment:

`EC_T_OS_PARMS::pfHwTimerGetInputFrequency`

`EC_T_OS_PARMS::pfHwTimerModifyInitialCount`

The master shift must not be enabled in the ENI file because it doesn't need any cyclic command. This mode can be activate using *emDcmConfigure()*

2.1.6 DCM Master Ref Clock mode

The DCM Master Ref Clock mode is similar to the bus shift mode, without its control loop. This reduces the CPU load and makes it a good alternative for low performance CPU. Because of the missing control loop, the reaction time on disturbance is longer and the cycle must be very accurate.

The bus shift time must be enabled in the ENI file because it use the same cyclic command to synchronize the EtherCAT network with master system This mode can be activate using *emDcmConfigure()*

2.1.7 DCM Linklayer Ref Clock mode

In this mode the link layer should provide the time base for the cyclic frames. `EC_LINKIOCTL_GETTIME` will be called during the DC initialization to initialize the DC related registers of the DC slaves and during the slave transition PREOP to SAFEOP to start the DC SYNC signals if needed.

`EC_LINKIOCTL_GETTIME` should return the current 64 bits value in nanosecond of a time counter running continuously.

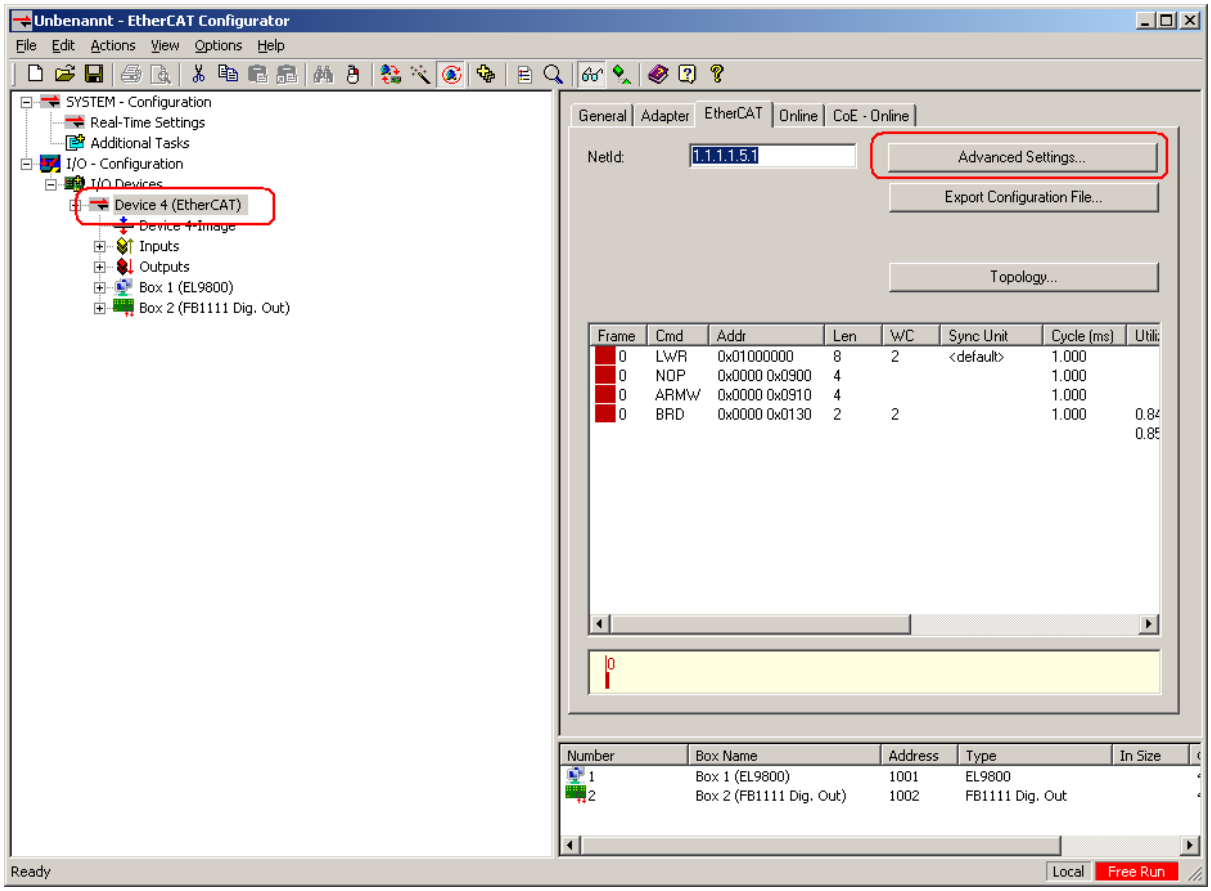
During the call to EcLinkSendFrame, the link layer should insert the send time of the frame following the instruction given by `EC_T_LINK_FRAMEDESC::wTimestampOffset` and `EC_T_LINK_FRAMEDESC::wTimestampSize` of the parameter pLinkFrameDesc. A value of 0 means that no time stamp should be inserted.

2.2 Configuration with ET9000

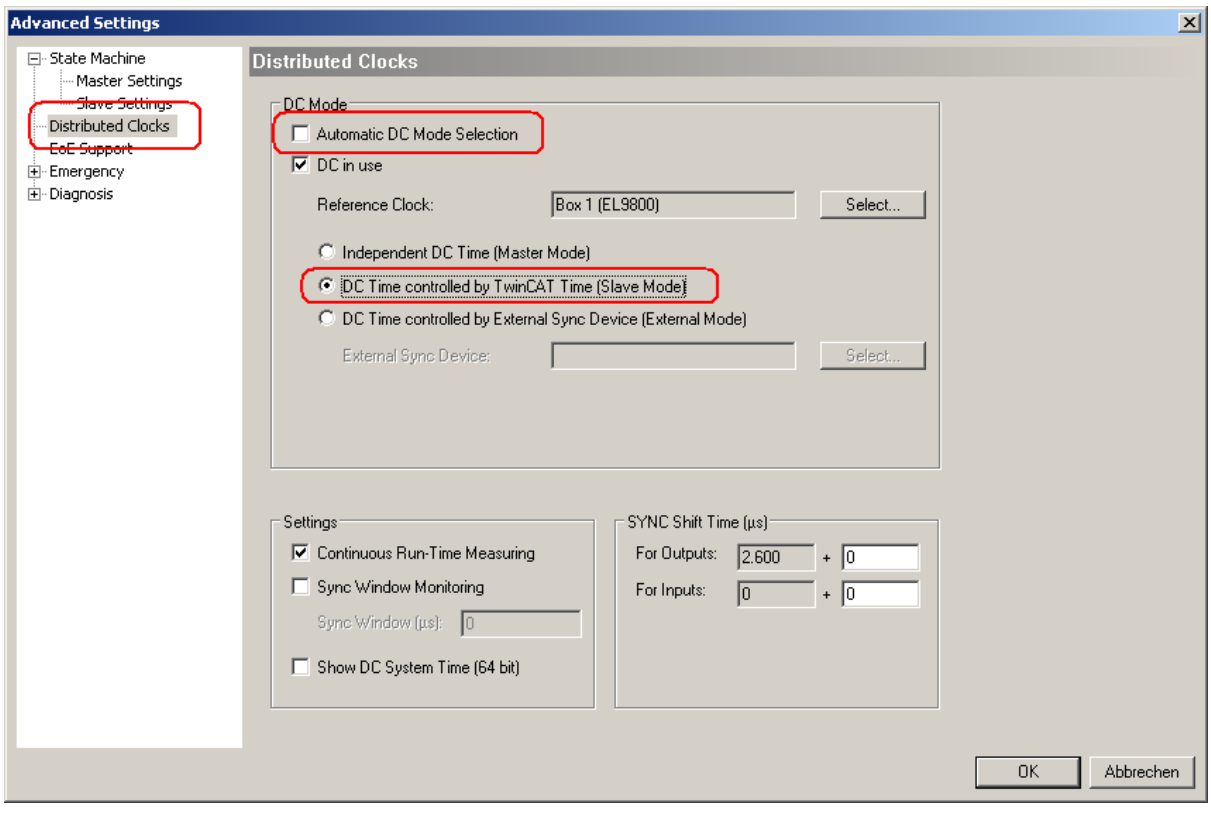
Since version 2.11.0 of the EtherCAT Configurator from Beckhoff explicitly setup whether the DC time shall be controlled by the EtherCAT master or not.

To create a DCM capable configuration, please accomplish the following steps.

1. Scan the EtherCAT Bus
2. **Select the EtherCAT device and press the button "Advanced Settings…"**



3. **In the open dialog please select “Distributed Clocks” on the left column. Then de-select “Automatic DC Mode Selection” and select the option “DC Time controlled by TwinCAT Time (Slave Mode)”.**



4. **Now the DC time can be controlled by the EtherCAT master. Don't forget to enable DC for the slaves.**

> **Note:** Don't forget to enable DC for the slaves.

2.3 Programmer's Guide

2.3.1 emDcmConfigure

static EC_T_DWORD **ecatDcmConfigure** (
    struct _EC_T_DCM_CONFIG *pDcmConfig,
    EC_T_DWORD dwInSyncTimeout
)

EC_T_DWORD **emDcmConfigure** (
    EC_T_DWORD dwInstanceID,
    EC_T_DCM_CONFIG *pDcmConfig,
    EC_T_DWORD dwInSyncTimeout
)

Configure DC master synchronization.

Parameters

- **dwInstanceID** – [in] Instance ID (Multiple EtherCAT Network Support)
- **pDcmConfig** – [in] Configuration information, a pointer to a structure of type *EC_T_DCM_CONFIG*.
- **dwInSyncTimeout** – [in] Currently not implemented.

Returns

EC_E_NOERROR or error code

struct **EC_T_DCM_CONFIG**

Public Members

EC_T_DCM_MODE **eMode**

[in] DCM mode

EC_T_DCM_CONFIG_BUSSHIFT **BusShift**

[in] BusShift configuration. Valid if eMode is set to eDcmMode_BusShift

EC_T_DCM_CONFIG_MASTERSHIFT **MasterShift**

[in] MasterShift configuration. Valid if eMode is set to eDcmMode_MasterShift

EC_T_DCM_CONFIG_LINKLAYERREFCLOCK **LinkLayerRefClock**

[in] LinkLayerRefClock configuration. Valid if eMode is set to eDcmMode_LinkLayerRefClock

EC_T_DCM_CONFIG_MASTERREFCLOCK **MasterRefClock**

[in] MasterRefClock configuration. Valid if eMode is set to eDcmMode_MasterRefClock

EC_T_DCM_CONFIG_DCX **Dcx**

[in] DCX configuration. Valid if eMode is set to eDcmMode_Dcx

enum **EC_T_DCM_MODE**

Values:

enumerator **eDcmMode_Off**
DCM disabled

enumerator **eDcmMode_BusShift**
DCM BusShift mode

enumerator **eDcmMode_MasterShift**
DCM MasterShift mode

enumerator **eDcmMode_LinkLayerRefClock**
DCM LinkLayer Ref Clock mode

enumerator **eDcmMode_MasterRefClock**
DCM Master Ref Clock mode

enumerator **eDcmMode_Dcx**
DCM DCX External synchronization mode

enumerator **eDcmMode_MasterShiftByApp**
DCM MasterShift controlled by application mode

struct **EC_T_DCM_CONFIG_BUSSHIFT**

Public Members

EC_T_INT **nCtlSetVal**
[in] Controller set value [ns]. This is the time distance between the cyclic frame send time and the DC base on bus (SYNC0 if shift is zero).

EC_T_INT **nCtlGain**
[in] Proportional gain in ppt (part per thousand). Default is value 2. A value of 0 let the current setting unmodified.

EC_T_INT **nCtlDriftErrorGain**
[in] Multiplier for drift error. Default value is 3. A value of 0 let the current setting unmodified

EC_T_INT **nMaxValidVal**
[in] Error inputs above this value are considered invalid. If error input prediction is valid then the difference between the error input and the expected value is taken. Default value is 3000. A value of 0 let the current setting unmodified

EC_T_BOOL **bLogEnabled**
[in] If set to EC_TRUE, logging information are generated and can be get calling emDcmGetLog

EC_T_DWORD **dwInSyncLimit**
[in] Limit [ns] for InSync monitoring. Default value is 20% of the cycle time. A value of 0 sets the default value.

EC_T_DWORD **dwInSyncSettleTime**
[in] Settle time [ms] for InSync monitoring. Default value is 1500ms. A value of 0 sets the default value.

EC_T_BOOL **bCtlOff**
[in] If set to EC_TRUE, control loop is disabled. Combined with bLogEnabled, it makes possible to analyze the natural drift between the stack cycle and the reference clock

EC_T_BOOL **bUseDcLoopCtlStdValues**
[in] If set to EC_TRUE, the values of ESC DC time loop control register 0x930 and 0x934 are not changed by master. This could increase the time it takes to get the InSync. Use only if there are a problems with the reference clock to get InSync

EC_T_DWORD **dwInSyncStartDelayCycle**
[in] Delay time [ms] before InSync monitoring start

EC_T_VOID ***pGetTimeElapsedSinceCycleStartContext**
[in] Optional context for the pfnGetTimeElapsedSinceCycleStart function, will be passed as first parameter to this function/static method.

EC_PF_DC_GETTIMEELAPSEDSINCECYCLESTART **pfnGetTimeElapsedSinceCycleStart**
[in] Setting this function enables synchronising the DC SYNC0 to the CycleStart instead of the time frame send. DC SYNC0 is therefore related to the CycleStart instead of the time frame send. Function must return EC_E_NOERROR to work. If 0 (nullptr, EC_NULL) EC_E_NOTSUPPORTED will be returned on attempt to call.

struct **EC_T_DCM_CONFIG_MASTERSHIFT**

Public Members

EC_T_INT **nCtlSetVal**
[in] Controller set value [ns]. This is the time distance between the cyclic frame send time and the DC base on bus (SYNC0 if shift is zero)

EC_T_INT **nCtlGain**
[in] Proportional gain in ppt (part per thousand). Default is value 2. A value of 0 let the current setting unmodified

EC_T_INT **nCtlDriftErrorGain**
[in] Multiplier for drift error. Default value is 3. A value of 0 let the current setting unmodified

EC_T_INT **nMaxValidVal**
[in] Error inputs above this value are considered invalid. If error input prediction is valid then the difference between the error input and the expected value is taken. Default value is 3000. A value of 0 let the current setting unmodified

EC_T_BOOL **bLogEnabled**
[in] If set to EC_TRUE, logging information are generated and can be get calling emDcmGetLog

EC_T_DWORD **dwInSyncLimit**
[in] Limit [ns] for InSync monitoring. Default value is 20% of the cycle time. A value of 0 sets the default value.

EC_T_DWORD **dwInSyncSettleTime**
[in] Settle time [ms] for InSync monitoring. Default value is 1500ms. A value of 0 sets the default value.

EC_T_BOOL **bCtlOff**
[in] If set to EC_TRUE, control loop is disabled. Combined with bLogEnabled, it makes possible to

analyze the natural drift between the stack cycle and the reference clock. Also it provides reading of current adjustment value using emDcmGetAdjust function

EC_T_DWORD **dwInSyncStartDelayCycle**
[in] Delay time [ms] before InSync monitoring start

EC_T_DC_STARTTIME_CB_DESC **DcStartTimeCallbackDesc**
[in] If not null, DC start time calculated by application, otherwise by master. See also *EC_T_DC_STARTTIME_CB_DESC*. Shift value configured in ENI will still be applied

EC_T_VOID ***pGetTimeElapsedSinceCycleStartContext**
[in] Optional context for the pfnGetTimeElapsedSinceCycleStart function, will be passed as first parameter to this function/static method.

EC_PF_DC_GETTIMEELAPSEDSINCECYCLESTART **pfnGetTimeElapsedSinceCycleStart**
[in] Setting this function enables synchronising the DC SYNC0 to the CycleStart instead of the time frame send. DC SYNC0 is therefore related to the CycleStart instead of the time frame send. Function must return EC_E_NOERROR to work. If 0 (nullptr, EC_NULL) EC_E_NOTSUPPORTED will be returned on attempt to call.

EC_T_VOID ***pAdjustCycleTimeContext**
[in] Optional context for the pfnAdjustCycleTime function, will be passed as first parameter to this function/static method.

EC_PF_DC_ADJUSTCYCLETIME **pfnAdjustCycleTime**
[in] Mandatory function to adjust the cycle time of the master. The master must adjust its cycle time by the passed amount to be in sync with the slaves. If 0 (nullptr, EC_NULL) MasterShift will not work.

struct **EC_T_DCM_CONFIG_LINKLAYERREFCLOCK**

Public Members

EC_T_INT **nCtlSetVal**
[in] Controller set value [ns]. This is the time distance between the cyclic frame send time and the DC base on bus (SYNC0 if shift is zero)

EC_T_BOOL **bLogEnabled**
[in] If set to EC_TRUE, logging information are generated and can be get calling emDcmGetLog

EC_T_DWORD **dwInSyncLimit**
[in] Limit [ns] for InSync monitoring. Default value is 20% of the cycle time. A value of 0 sets the default value.

EC_T_DWORD **dwInSyncSettleTime**
[in] Settle time [ms] for InSync monitoring. Default value is 1500ms. A value of 0 sets the default value.

EC_T_DC_STARTTIME_CB_DESC **DcStartTimeCallbackDesc**
[in] If not null, DC start time calculated by application, otherwise by master. See also *EC_T_DC_STARTTIME_CB_DESC*. Shift value configured in ENI will still be applied.

struct **EC_T_DCM_CONFIG_MASTERREFCLOCK**

Public Members

EC_T_INT **nCtlSetVal**
[in] Controller set value [ns]. This is the time distance between the cyclic frame send time and the DC base on bus (SYNC0 if shift is zero)

EC_T_BOOL **bLogEnabled**
[in] If set to EC_TRUE, logging information are generated and can be get calling emDcmGetLog

EC_T_DWORD **dwInSyncLimit**
[in] Limit [ns] for InSync monitoring. Default value is 20% of the cycle time. A value of 0 sets the default value.

EC_T_DWORD **dwInSyncSettleTime**
[in] Settle time [ms] for InSync monitoring. Default value is 1500ms. A value of 0 sets the default value.

EC_T_VOID *__pGetHostTimeContext__
[in] Optional context for the pfnGetHostTime function, will be passed as first parameter to this function/static method.

EC_PF_DC_GETHOSTTIME **pfnGetHostTime**
[in] Mandatory function to offer the host time for the MasterRefClock mode. This time will be distributed to the slaves. If 0 (nullptr, EC_NULL) MasterRefClock mode will not work.

struct **EC_T_DCM_CONFIG_DCX**

Contains the configuration information for the DCX external synchronization mode.

See also:

Feature Pack “External Synchronization” for further details.

struct **EC_T_DC_STARTTIME_CB_DESC**

Public Members

EC_T_VOID *__pvContext__
[in] Context pointer. It is used as parameter when the callback function is called

EC_PF_DC_STARTTIME_CB **pfnCallback**
[in] Dc start time callback function pointer. If not null, DC start time calculated by application, otherwise by master

typedef EC_T_DWORD (***EC_PF_DC_STARTTIME_CB**)(EC_T_VOID *pvContext, EC_T_WORD wSlaveFixedAddr, EC_T_UINT64 *pqwDcStartTime)
EC-Master requests DC start time for every single slave from a given callback DcStartTimeCallbackDesc with slave station address as input parameter. The slave specific DC start time value will be passed directly to the slave without modifications by master. This means no other values like nCtlSetVal will be added. Shift value configured in ENI will still be applied.

Parameters

- **pvContext** – [in] Context pointer. It is used as parameter when the callback function is called.
- **wSlaveFixedAddr** – [in] Slave fixed address.

- **pqwDcStartTime** – [out] DC start time for specific slave.

Returns
EC_E_NOERROR or error code

2.3.2 emDcmGetStatus

static EC_T_DWORD **ecatDcmGetStatus** (
EC_T_DWORD *pdwErrorCode,
EC_T_INT *pnDiffCur,
EC_T_INT *pnDiffAvg,
EC_T_INT *pnDiffMax
)
EC_T_DWORD **emDcmGetStatus** (
EC_T_DWORD dwInstanceID,
EC_T_DWORD *pdwErrorCode,
EC_T_INT *pnDiffCur,
EC_T_INT *pnDiffAvg,
EC_T_INT *pnDiffMax
)
Get DC master synchronization controller status.

Parameters

- **dwInstanceID** – [in] Instance ID (Multiple EtherCAT Network Support)
- **pdwErrorCode** – [out] Pointer to current error code of the DCM controller. Possible values are:
  - EC_E_NOTREADY DCM control loop is not running
  - EC_E_BUSY DCM control loop is running and try to get InSync
  - *DCM_E_MAX_CTL_ERROR_EXCEED* Set if the controller error exceeds the InSyncLimit
  - *DCM_E_DRIFT_TO_HIGH* DCM control loop not able to compensate drift. Drift above 600ppm.
- **pnDiffCur** – [out] Pointer to current difference between set value and actual value of controller in nanoseconds.
- **pnDiffAvg** – [out] Pointer to average difference between set value and actual value of controller in nanoseconds
- **pnDiffMax** – [out] Pointer to maximum difference between set value and actual value of controller in nanoseconds

Returns

- EC_E_NOERROR if status retrieval was successful
- EC_E_NOTSUPPORTED the DC feature is not supported/switched off. EC-Master stack has to be compiled with DC support see “define INCLUDE_DC_SUPPORT”
- EC_NULL does not appear in normal flow, if EC_NULL is returned we are in a very exceptional case where m_poDcm == NULL (SW error detection)

2.3.3 emDcmResetStatus

static EC_T_DWORD **ecatDcmResetStatus**(EC_T_VOID)

EC_T_DWORD **emDcmResetStatus**(EC_T_DWORD dwInstanceID)

Reset DC master synchronization controller status, average and maximum difference between set value and actual value.

Parameters

dwInstanceID – [in] Instance ID (Multiple EtherCAT Network Support)

Returns

EC_E_NOERROR or error code

2.3.4 emDcmGetBusShiftConfigured

static EC_T_DWORD **ecatDcmGetBusShiftConfigured**(EC_T_BOOL *pbBusShiftConfigured)

EC_T_DWORD **emDcmGetBusShiftConfigured**(
EC_T_DWORD dwInstanceID,
EC_T_BOOL *pbBusShiftConfigured
)

Determines if DCM Bus Shift is configured/possible in configuration (ENI file)

Parameters

- **dwInstanceID** – [in] Instance ID (Multiple EtherCAT Network Support)
- **pbBusShiftConfigured** – [out] EC_TRUE if DCM bus shift mode is supported by the current configuration

Returns

EC_E_NOERROR or error code

2.3.5 emDcmGetLog

static EC_T_DWORD **ecatDcmGetLog**(EC_T_CHAR **pszLog)

EC_T_DWORD **emDcmGetLog**(EC_T_DWORD dwInstanceID, EC_T_CHAR **pszLog)

Get logging information from the DCM controller.

This function returns non-zero pointer only if bLogEnabled was set to EC_TRUE in *EC_T_DCM_CONFIG*.

Parameters

- **dwInstanceID** – [in] Instance ID (Multiple EtherCAT Network Support)
- **pszLog** – [out] Pointer to a string containing the current logging information

Returns

EC_E_NOERROR or error code

See also:

Controller adjustment for content description of pszLog

2.3.6 emIoControl - EC_IOCTL_DCM_GET_LOG

Get logging information from the DCM controller.

emIoControl - EC_IOCTL_DCM_GET_LOG

Parameter

- `pbyInBuf`: [in] Should be set to EC_NULL
- `dwInBufSize`: [in] Should be set to 0
- `pbyOutBuf`: [out] Pointer to struct EC_T_DCM_LOG
- `dwOutBufSize`: [in] Size of the output buffer in bytes
- `pdwNumOutData`: [out] Pointer to EC_T_DWORD. Amount of bytes written to the output buffer pbyOutBuf

Return
EC_E_NOERROR or error code

struct **EC_T_DCM_LOG**

Public Members

EC_T_DWORD **dwMsecCounter**
[out] Current MsecCounter

EC_T_INT **nCtlSetVal**
[out] Configured controller set val [ns]

EC_T_UINT64 **qwBusTime**
[out] Current BusTime

EC_T_INT **nCtlErrorNsec**
[out] Current controller error [ns]

EC_T_INT **nDrift**
[out] Current calculated drift [ppm]

EC_T_DWORD **dwErrorCode**
[out] Last returned error code by controller

EC_T_BOOL **bDcmInSync**
[out] EC_TRUE if DCM is in sync, EC_FALSE if out of sync

EC_T_BOOL **bDcInSync**
[out] EC_TRUE if DC is in sync, EC_FALSE if out of sync

EC_T_UINT64 **qwDcStartTime**
[out] Last used DC StartTime

EC_T_INT **nSystemTimeDifference**
[out] Last read System Time Difference (ESC register 0x092C)

2.3.7 emDcmShowStatus

static EC_T_DWORD **ecatDcmShowStatus**(EC_T_VOID)

EC_T_DWORD **emDcmShowStatus**(EC_T_DWORD dwInstanceID)
Show DC master synchronization status as DbgMsg (for development purposes only).

Parameters
dwInstanceID – [in] Instance ID (Multiple EtherCAT Network Support)

Returns
EC_E_NOERROR or error code

2.3.8 emDcmGetAdjust

static EC_T_DWORD **ecatDcmGetAdjust**(EC_T_INT *pnAdjustPermil)

EC_T_DWORD **emDcmGetAdjust**(EC_T_DWORD dwInstanceID, EC_T_INT *pnAdjustPermil)
Returns the current adjustment value for the timer.

bCtlOff must be set to EC_TRUE in *EC_T_DCM_CONFIG* to enable external adjustment.

Parameters

- **dwInstanceID** – [in] Instance ID (Multiple EtherCAT Network Support)
- **pnAdjustPermil** – [out] Current adjustment value of the timer.

Returns
EC_E_NOERROR or error code

2.3.9 DCM specific error codes

DCM_E_ERROR
0x981201C0: Unspecific DCM Error

DCM_E_NOTINITIALIZED
0x981201C1: Not initialized

DCM_E_MAX_CTL_ERROR_EXCEED
0x981201C2: DCM controller - synchronization out of limit

DCM_E_NOMEMORY
0x981201C3: Not enough memory

DCM_E_INVALID_HWLAYER
0x981201C4: Hardware layer - (BSP) invalid

DCM_E_TIMER_MODIFY_ERROR
0x981201C5: Hardware layer - error modifying timer

DCM_E_TIMER_NOT_RUNNING
0x981201C6: Hardware layer - timer not running

DCM_E_WRONG_CPU
0x981201C7: Hardware layer - function called on wrong CPU

DCM_E_INVALID_SYNC_PERIOD
0x981201C8: Invalid DC sync period length (invalid clock master?)

DCM_E_INVALID_SETVAL
0x981201C9: DCM controller SetVal to small

DCM_E_DRIFT_TO_HIGH
0x981201CA: DCM controller - Drift between local timer and ref clock to high

DCM_E_BUS_CYCLE_WRONG
0x981201CB: DCM controller - Bus cycle time (dwBusCycleTimeUsec) doesn't match real cycle

DCX_E_NO_EXT_CLOCK
0x981201CC: DCX controller - No external synchronization clock found

DCM_E_INVALID_DATA
0x981201CD: DCM controller - Invalid data

2.3.10 Notifications

At startup the master raises the notifications *emNotify - EC_NOTIFY_DC_SLV_SYNC*, *emNotify - EC_NOTIFY_DC_STATUS* and *emNotify - EC_NOTIFY_DCM_SYNC* at master state transition from INIT to PREOP.

The order is typically as follows (*emNotify - EC_NOTIFY_DCM_SYNC* may be before or after reaching PREOP):

EC_NOTIFY_STATECHANGED(INIT)[…]

EC_NOTIFY_DC_SLV_SYNC […]

EC_NOTIFY_DC_STATUS […]

[EC_NOTIFY_DCM_SYNC] […]

EC_NOTIFY_STATECHANGED(PREOP) […]

[EC_NOTIFY_DCM_SYNC] […]

EC_NOTIFY_STATECHANGED(SAFEOP) […]

EC_NOTIFY_STATECHANGED(OP) […]

2.3.11 emNotify - EC_NOTIFY_DCM_SYNC

DCM InSync notification.

This notification is enabled by default.

emNotify - EC_NOTIFY_DCM_SYNC

Parameter

- `pbyInBuf`: [in] pointer to notification descriptor EC_T_DCM_SYNC_NTFY_DESC
- `dwInBufSize`: [in] sizeof(EC_T_DCM_SYNC_NTFY_DESC).
- `pbyOutBuf`: [out] Should be set to EC_NULL
- `dwOutBufSize`: [in] Should be set to 0
- `pdwNumOutData`: [out] Should be set to EC_NULL

struct **EC_T_DCM_SYNC_NTFY_DESC**

Public Members

EC_T_DWORD **IsInSync**
: [in] EC_TRUE as long as time of master and reference clock are in sync. False if the InSyncLimit from the bus shift configuration is exceeded

EC_T_INT **nCtlErrorNsecCur**
: [in] Current difference [ns] between set value and actual value of controller

EC_T_INT **nCtlErrorNsecAvg**
: [in] Average difference [ns] between set value and actual value of controller

EC_T_INT **nCtlErrorNsecMax**
: [in] Maximum difference [ns] between set value and actual value of controller

See also:

emIoControl - EC_IOCTL_SET_NOTIFICATION_ENABLED in the EC-Master Class B documentation for how controls the deactivation

2.4 Example codes

A master application which includes DCM API needs to call following steps:

Main Thread:

Master initialization and controller configuration, main loop, shutdown.

```
#include <EcMaster.h>

/* initialize the master */
dwRes = emInitMaster(&oInitParms);

/* configure the master */
dwRes = emConfigureNetwork(eCnfType_Filename, (EC_T_PBYTE)szCfgFile,␣
↪OsStrlen(szCfgFile));

/* register client */
dwRes = emRegisterClient(emNotifyCallback, pNotification, &oRegisterResults);

/* configure DCM bus shift */
OsMemset(&oDcmConfigure, 0, sizeof(EC_T_DCM_CONFIG_BUSSHIFT));
oDcmConfigure.nCtlSetVal  = DCM_CONTROLLER_SETVAL_NANOSEC;
oDcmConfigure.bLogEnabled = EC_FALSE;
dwRes = emDcmConfigure(&oDcmConfigure, 0);

/* set master and bus state to OP */
dwRes = emSetMasterState(dwStartTimeout+dwScanBustimeout, eemState_OP);

/* application loop */
while(bRun)
{
    dwRes = emDcmGetStatus(&dwStatus, &nDiffCur, &nDiffAvg, &nDiffMax);
}
```

(continues on next page)

(continued from previous page)

```
/* stop master operation */
dwTmpRes = emStop(dwStartTimeout);
```

Cyclic (Job) Thread:

Master jobs and dcm logging.

```
EC_T_VOID tEcJobTask(EC_T_VOID* pvAppThreadParamDesc)
{
    while (!pDemoThreadParam->bJobThreadShutdown)
        {
                dwRes = emExecJob(eUsrJob_ProcessAllRxFrames, &bPrevCycProcessed);

                /* get logging information */
                dwRes = emDcmGetLog(&pszLog);

                /* send all cyclic frames */
                dwRes = emExecJob(eUsrJob_SendAllCycFrames, EC_NULL);

                /* run the master timer handler */
                dwRes = emExecJob(eUsrJob_MasterTimer, EC_NULL);

                /* send all queued acyclic EtherCAT frames */
                dwRes = emExecJob(eUsrJob_SendAcycFrames, EC_NULL);

                OsSleep(CYCLE_TIME);
        }
}
```

For closer details find a DCM example in project “EcMasterDemoDc” in the folder “Examples”.

3 Running EcMasterDemoDc

The EcMasterDemoDc is available “out of the box” for different operating systems. It is an EC-Master example application that handles the following tasks:

- Showing basic EtherCAT communication
- Master stack initialization into OPERATIONAL state
- DC and DCM configuration
- Process Data operations for e.g. Beckhoff EL2004, EL1004 and EL4132
- Periodic diagnosis task
- Periodic Job Task in polling mode
- Logging

Start the EcMasterDemoDc from the command line to put the EtherCAT network into operation. At least a Link Layer must be specified.

```
> EcMasterDemoDc -winpcap 192.168.157.2 1 -f eni.xml -t 0 -v 3 -dcmmode busshift
```

3.1 Command line parameters

EcMasterDemoDc <LinkLayer> [-f ENI-FileName] [-t time] [-b cycle time] [-a affinity] [-v level] [-perf [level]] [-log prefix [msg cnt]] [-lic key] [-oem key] [-maxbusslaves cnt] [-flash address] [-sp [port]] [-dcmmode mode] [-ctloff] [-rec [prefix [frame cnt]]]

The parameters are as follows:

-f <configFileName>
: Path to ENI file

-t <time>
: Running duration in msec. When the time expires the demo application exits completely.

  <time>
  : Time in msec, 0 = forever (default = 120000)

-b <cycle time>
: Specifies the bus cycle time. Defaults to 1000 µs (1 ms).

  <cycle time>
  : Bus cycle time in µsec

-a <affinity>
: The CPU affinity specifies which CPU the demo application ought to use.

  <affinity>
  : 0 = first CPU, 1 = second, …

-v <level>
: The verbosity level specifies how much console output messages will be generated by the demo application. A high verbosity level leads to more messages.

  <level>
  : Verbosity level: 0=off (default), 1..n=more messages

-perf [<level>]
: Enable max. and average time measurement in µs for all EtherCAT jobs (e.g. ProcessAllRxFrames).

<level>
Depending on level the performance histogram can be activated as well.

-log <prefix> [<msg cnt>]
Use given file name prefix for log files.

<prefix>

<msg cnt>
Messages count for log buffer allocation

-lic <key>
Use License key.

<key>
26 characters long license key.

-oem <key>
Use OEM key

<key>
64 bit OEM key.

-maxbusslaves <cnt>
Set max number of slaves

-flash <address>
Flash outputs

<address>
0=all, >0 = slave station address

-sp [<port>]
If platform has support for IP Sockets, this command-line option enables the Remote API Server to be started. The Remote API Server is going to listen on TCP Port 6000 (or port parameter if given) and is available for connecting Remote API Clients.

<port>
RAS server port

-rec [<prefix> [<frame cnt>]]
Packet capture file recording

<prefix>
File name prefix

<frame cnt>
Frame count for log buffer allocation

-dcmmode <mode>
Set DCM mode

<mode>
off | busshift | mastershift | masterrefclock | linklayerrefclock

-ctloff
Disable DCM control loop for diagnosis