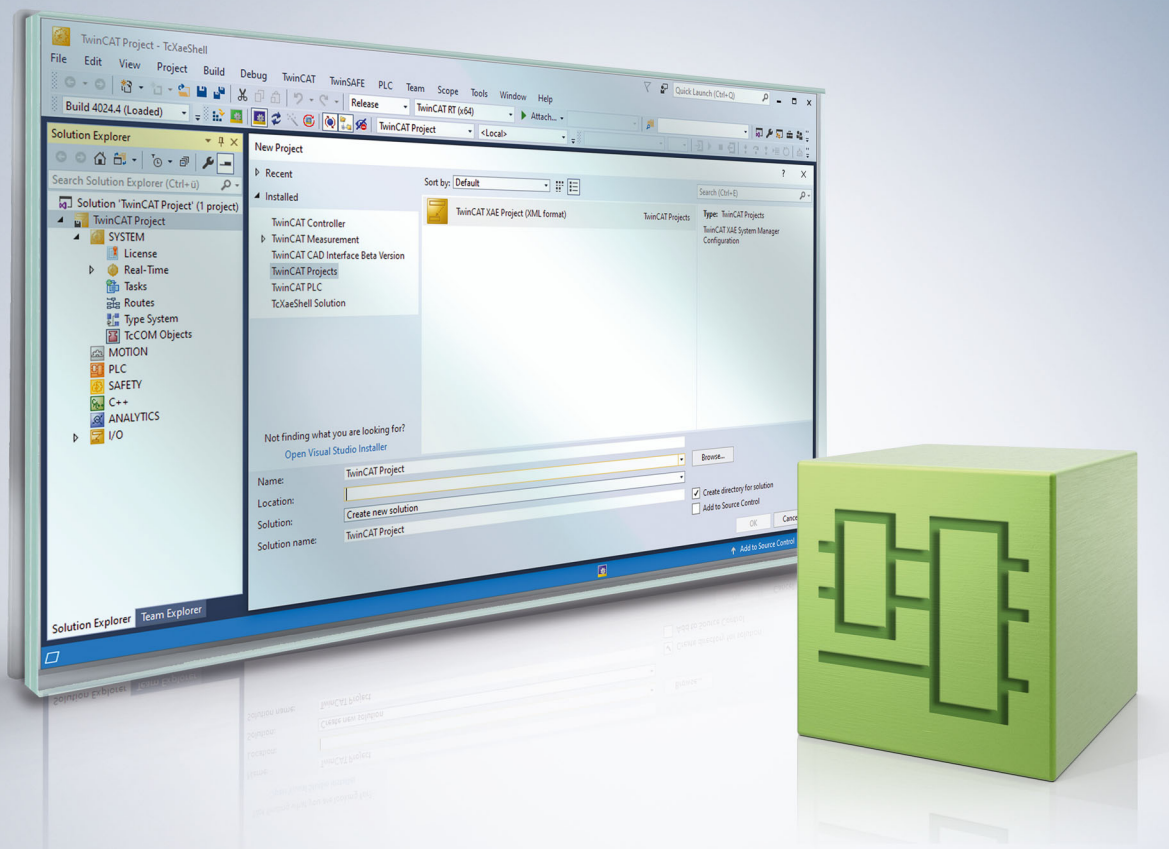
BECKHOFF New Automation Technology

Handbuch | DE

TE1000

TwinCAT 3 | PLC-Bibliothek: Tc3_EtherCATDiag



10.07.2024 | Version: 1.0.0

Inhaltsverzeichnis

1 Vorwort

1.1 Hinweise zur Dokumentation

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs- und Automatisierungstechnik, das mit den geltenden nationalen Normen vertraut ist.
Zur Installation und Inbetriebnahme der Komponenten ist die Beachtung der Dokumentation und der nachfolgenden Hinweise und Erklärungen unbedingt notwendig.
Das Fachpersonal ist verpflichtet, stets die aktuell gültige Dokumentation zu verwenden.

Das Fachpersonal hat sicherzustellen, dass die Anwendung bzw. der Einsatz der beschriebenen Produkte alle Sicherheitsanforderungen, einschließlich sämtlicher anwendbaren Gesetze, Vorschriften, Bestimmungen und Normen erfüllt.

Disclaimer

Diese Dokumentation wurde sorgfältig erstellt. Die beschriebenen Produkte werden jedoch ständig weiterentwickelt.
Wir behalten uns das Recht vor, die Dokumentation jederzeit und ohne Ankündigung zu überarbeiten und zu ändern.
Aus den Angaben, Abbildungen und Beschreibungen in dieser Dokumentation können keine Ansprüche auf Änderung bereits gelieferter Produkte geltend gemacht werden.

Marken

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® und XPlanar® sind eingetragene und lizenzierte Marken der Beckhoff Automation GmbH.
Die Verwendung anderer in dieser Dokumentation enthaltenen Marken oder Kennzeichen durch Dritte kann zu einer Verletzung von Rechten der Inhaber der entsprechenden Bezeichnungen führen.

Patente

Die EtherCAT-Technologie ist patentrechtlich geschützt, insbesondere durch folgende Anmeldungen und Patente:
EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
mit den entsprechenden Anmeldungen und Eintragungen in verschiedenen anderen Ländern.

EtherCAT® ist eine eingetragene Marke und patentierte Technologie lizenziert durch die Beckhoff Automation GmbH, Deutschland

Copyright

1.2 Zu Ihrer Sicherheit

Sicherheitsbestimmungen

Lesen Sie die folgenden Erklärungen zu Ihrer Sicherheit.
Beachten und befolgen Sie stets produktspezifische Sicherheitshinweise, die Sie gegebenenfalls an den entsprechenden Stellen in diesem Dokument vorfinden.

Haftungsausschluss

Die gesamten Komponenten werden je nach Anwendungsbestimmungen in bestimmten Hard- und Software-Konfigurationen ausgeliefert. Änderungen der Hard- oder Software-Konfiguration, die über die dokumentierten Möglichkeiten hinausgehen, sind unzulässig und bewirken den Haftungsausschluss der Beckhoff Automation GmbH & Co. KG.

Qualifikation des Personals

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs-, Automatisierungs- und Antriebstechnik, das mit den geltenden Normen vertraut ist.

Signalwörter

Im Folgenden werden die Signalwörter eingeordnet, die in der Dokumentation verwendet werden. Um Personen- und Sachschäden zu vermeiden, lesen und befolgen Sie die Sicherheits- und Warnhinweise.

Warnungen vor Personenschäden

⚠ GEFAHR

Es besteht eine Gefährdung mit hohem Risikograd, die den Tod oder eine schwere Verletzung zur Folge hat.

⚠ WARNUNG

Es besteht eine Gefährdung mit mittlerem Risikograd, die den Tod oder eine schwere Verletzung zur Folge haben kann.

⚠ VORSICHT

Es besteht eine Gefährdung mit geringem Risikograd, die eine mittelschwere oder leichte Verletzung zur Folge haben kann.

Warnung vor Umwelt- oder Sachschäden

HINWEIS

Es besteht eine mögliche Schädigung für Umwelt, Geräte oder Daten.

Information zum Umgang mit dem Produkt

Diese Information beinhaltet z. B.:
Handlungsempfehlungen, Hilfestellungen oder weiterführende Informationen zum Produkt.

1.3 Hinweise zur Informationssicherheit

Die Produkte der Beckhoff Automation GmbH & Co. KG (Beckhoff) sind, sofern sie online zu erreichen sind, mit Security-Funktionen ausgestattet, die den sicheren Betrieb von Anlagen, Systemen, Maschinen und Netzwerken unterstützen. Trotz der Security-Funktionen sind die Erstellung, Implementierung und ständige Aktualisierung eines ganzheitlichen Security-Konzepts für den Betrieb notwendig, um die jeweilige Anlage, das System, die Maschine und die Netzwerke gegen Cyber-Bedrohungen zu schützen. Die von Beckhoff verkauften Produkte bilden dabei nur einen Teil des gesamtheitlichen Security-Konzepts. Der Kunde ist dafür verantwortlich, dass unbefugte Zugriffe durch Dritte auf seine Anlagen, Systeme, Maschinen und Netzwerke verhindert werden. Letztere sollten nur mit dem Unternehmensnetzwerk oder dem Internet verbunden werden, wenn entsprechende Schutzmaßnahmen eingerichtet wurden.

Zusätzlich sollten die Empfehlungen von Beckhoff zu entsprechenden Schutzmaßnahmen beachtet werden. Weiterführende Informationen über Informationssicherheit und Industrial Security finden Sie in unserem https://www.beckhoff.de/secguide.

Die Produkte und Lösungen von Beckhoff werden ständig weiterentwickelt. Dies betrifft auch die Security-Funktionen. Aufgrund der stetigen Weiterentwicklung empfiehlt Beckhoff ausdrücklich, die Produkte ständig auf dem aktuellen Stand zu halten und nach Bereitstellung von Updates diese auf die Produkte aufzuspielen. Die Verwendung veralteter oder nicht mehr unterstützter Produktversionen kann das Risiko von Cyber-Bedrohungen erhöhen.

Um stets über Hinweise zur Informationssicherheit zu Produkten von Beckhoff informiert zu sein, abonnieren Sie den RSS Feed unter https://www.beckhoff.de/secinfo.

2 Übersicht

Die SPS-Bibliothek Tc3_EtherCATDiag wird zur erweiterten Diagnose von EtherCAT Slaves genutzt.

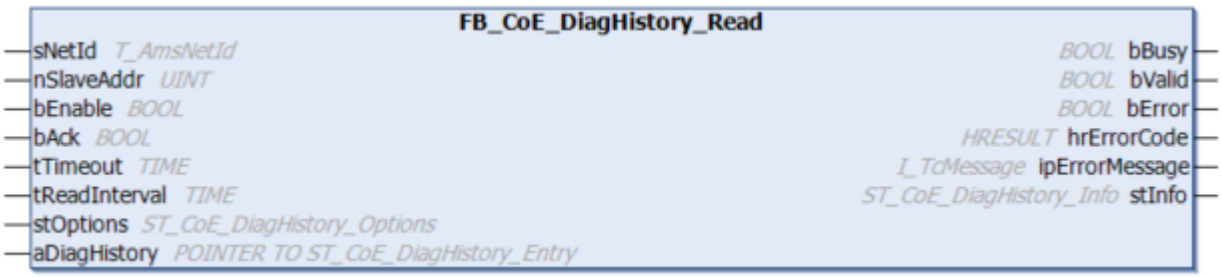
Mit den Funktionsbausteinen FB_ReadCoEDiagHistory und FB_ReadCoEDiagHistory_Axis können EtherCAT Slaves, die das CoE Diag History Object unterstützen, ausgelesen werden.

Systemvoraussetzung

Target System	Win10, Win11 IPC or CX, (x86, x64, ARM)
Min. TwinCAT-Version	3.1.4024.57
Min. TwinCAT-Level	TC1200 TC3 PLC

3 Funktionsbausteine

3.1 FB_CoE_DiagHistory_Read



Der Funktionsbaustein FB_ReadCoEDiagHistory kann EtherCAT Slaves, die das CoE Diag History Object unterstützen, auslesen. Der EtherCAT Slave muss sich dafür an die ETG-Norm halten.

Syntax

```
VAR_INPUT
    sNetId          :  T_AmsNetId;
    nSlaveAddr      :  UINT;
    bEnable         :  BOOL;
    bAck            :  BOOL;
    tTimeout        :  TIME := DEFAULT_ADS_TIMEOUT;
    tReadInterval   :  TIME := T#10S;
    stOptions       :  ST_CoE_DiagHistory_Options;
END_VAR
VAR_IN_OUT
    aDiagHistory    :  ARRAY[*] OF ST_CoE_DiagHistory_Entry;
END_VAR
VAR_OUTPUT
    bBusy           :  BOOL;
    bValid          :  BOOL;
    bError          :  BOOL;
    hrErrorCode     :  HRESULT;
    ipErrorMessage  :  I_TcMessage := fbResult;
    stInfo          :  ST_CoE_DiagHistory_Info;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
sNetId	T_AmsNetId	NetID des Slaves
nSlaveAddr	UINT	Port-Adresse des Slaves
bEnable	BOOL	Puls: die Diag History wird einmal ausgelesen. Dauerhaft: die Diag History wird wiederholt ausgelesen (siehe tReadInterval). Bei jeder neuen Flanke, wird alles resetet. Anschließend wird die Verbindung neu aufgebaut und der Slave ausgelesen.
bAck	BOOL	Alle Meldungen werden quittiert, wenn bAck getriggert wird.
tTimeout	TIME	Timeout für die jeweiligen ADS Reads/Writes im Baustein. Default ist hier ADS-Timeout von fünf Sekunden.
tReadInterval	TIME	Zeitintervall zwischen dem wiederholten Auslesen des Slaves bei dauerhaftem „bEnable“-Signal. Default ist ein Zeitintervall von 10 Sekunden.
stOptions	ST_CoE_DiagHistory_Options [▶ 14]	Steuerelemente zur Handhabung von CoE Diag History

Ein-/Ausgänge

Name	Typ	Beschreibung
aDiagHistory	ARRAY[*] OF ST_CoE_DiagHistory_Entry [▶ 13]	Dieser In-Out-Variable wird ein Array beliebiger Länge von ST_CoE_DiagHistory_Entry zugewiesen. Der FB passt sich automatisch der Länge des Arrays an. Wenn es zu kurz ist, werden ältere Meldungen im Diag History Object des Slaves ignoriert. Wenn es zu lang ist, werden überflüssige Einträge gelöscht. Eine größere Länge als 250 ergibt hier keinen Sinn, da die ETG-Norm ein Max. von 250 Meldungen für das Diag History Object definiert.

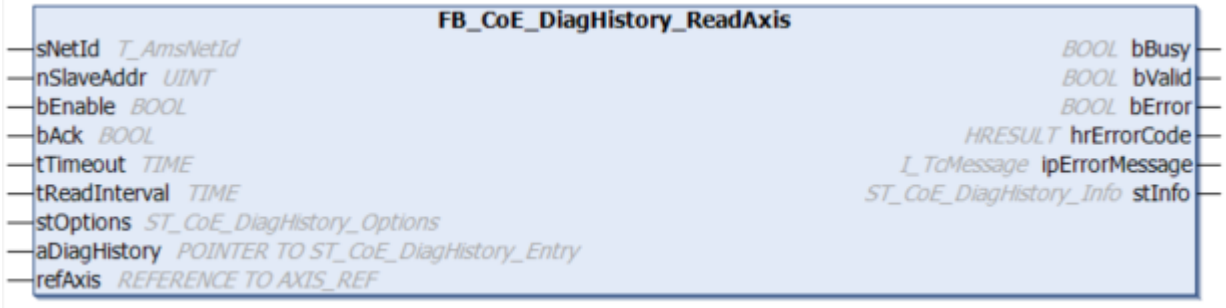
Ausgänge

Name	Typ	Beschreibung
bBusy	BOOL	Einfaches Busy Signal, es signalisiert, dass der Baustein einen Slave ausliest.
bValid	BOOL	Das Diag History Object wurde erfolgreich ausgelesen und die ausgegebenen Meldungen sind gültig.
bError	BOOL	Wird TRUE, sobald eine Fehlersituation eintritt.
hrErrorCode	HRESULT	Liefert bei einem gesetzten bError-Ausgang einen Fehlercode.
ipErrorMessage	I_TcMessage	Liefert bei einem gesetzten bError-Ausgang detaillierte Informationen. Der hierzu verwendete Schnittstellenzeiger ist immer gültig (ungleich Null) und vom Typ I_TcMessage. Insbesondere ist der entsprechende Fehler im PLC-OnlineView sofort als Klartext zu sehen.
stInfo	ST_CoE_DiagHistory_Info [▶ 13]	Liefert Informationen vom ausgelesenen EtherCAT Slave Diag History Object. Dies beinhaltet u.a. den verwendeten Buffer Mode und Informationen zu den verfügbaren Meldungen. Dieser Ausgang wird nur nach erfolgreichem Lesen aktualisiert.

Voraussetzungen

TwinCAT Version	Hardware	Einzubindende Bibliotheken
TwinCAT 3.1 >= Build 4024.57	x86, x64, ARM	Tc3_EtherCATDiag

3.2 FB_CoE_DiagHistory_ReadAxis



Der Funktionsbaustein FB_ReadCoEDiagHistory_Axis ist eine Variation des Funktionsbausteins FB_ReadCoEDiagHistory [▶ 9], dem an Stelle einer Adresse eine AXIS_REF-Struktur zugewiesen werden kann. Es werden nur Meldungen ausgegeben, die entweder zum Gerät oder zu der spezifischen Achse gehören (bspw. bei einem AX8206 Doppelachsmodul).

Syntax

```
VAR_INPUT
    sNetId           :  T_AmsNetId;
    nSlaveAddr       :  UINT;
    bEnable          :  BOOL;
    bAck             :  BOOL;
    tTimeout         :  TIME := DEFAULT_ADS_TIMEOUT;
    tReadInterval    :  TIME := T#10S;
    stOptions        :  ST_CoE_DiagHistory_Options;
END_VAR
VAR_IN_OUT
    aDiagHistory     :  ARRAY[*] OF ST_CoE_DiagHistoryEntry;
END_VAR
VAR_INPUT
    refAxis          :  REFERNCE TO AXIS_REF;
END_VAR
VAR_OUTPUT
    bBusy            :  BOOL;
    bValid           :  BOOL;
    bError           :  BOOL;
    hrErrorCode      :  HRESULT;
    ipErrorMessage   :  I_TcMessage := fbResult;
    stInfo           :  ST_CoE_DiagHistory_Info;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
sNetId	T_AmsNetId	Dieser Eingang wird nicht verwendet und sollte nicht zugewiesen werden.
nSlaveAddr	UINT	Dieser Eingang wird nicht verwendet und sollte nicht zugewiesen werden.
bEnable	BOOL	Puls: die Diag History wird einmal ausgelesen. Dauerhaft: die Diag History wird wiederholt ausgelesen (siehe tReadInterval). Bei jeder neuen Flanke, wird erst alles resetet. Anschließend wird die Verbindung neu aufgebaut und der Slave ausgelesen.
bAck	BOOL	Alle Meldungen werden quittiert, wenn bAck getriggert wird.
tTimeout	TIME	Timeout für die jeweiligen ADS Reads/Writes im Baustein. Default ist hier der ADS-Timeout von fünf Sekunden.
tReadInterval	TIME	Zeitintervall zwischen dem wiederholten Auslesen des Slaves bei dauerhaftem „bEnable“-Signal. Default ist ein Zeitintervall von 10 Sekunden.
stOptions	ST_CoE_DiagHistory_Options [▶ 14]	Steuerelemente zur Handhabung von CoE Diag History
refAxis	REFERNCE TO AXIS_REF	AXIS_REF einer Achse, aus dieser kann die Adresse des entsprechenden Slaves ermittelt werden.

Ein-/Ausgänge

Name	Typ	Beschreibung
aDiagHistory	ARRAY[*] OF ST_CoE_DiagHistory_Entry [▶ 13]	Dieser In-Out-Variable wird ein Array beliebiger Länge von ST_CoE_DiagHistory_Entry zugewiesen. Der FB passt sich automatisch der Länge des Arrays an. Wenn es zu kurz ist, werden ältere Meldungen im Diag History Object des Slaves ignoriert. Wenn es zu lang ist, werden überflüssige Einträge gelöscht. Eine größere Länge als 250 ergibt hier keinen Sinn, da die ETG-Norm ein Max. von 250 Meldungen für das Diag History Object definiert.

Ausgänge

Name	Typ	Beschreibung
bBusy	BOOL	Einfaches Busy Signal, signalisiert, dass der Baustein einen Slave ausliest.
bValid	BOOL	Das Diag History Object wurde erfolgreich ausgelesen und die ausgegebenen Meldungen sind gültig.
bError	BOOL	Wird TRUE, sobald eine Fehlersituation eintritt.
hrErrorCode	HRESULT	Liefert bei einem gesetzten bError-Ausgang einen Fehlercode.
ipErrorMessage	I_TcMessage	Liefert bei einem gesetzten bError-Ausgang detaillierte Informationen. Der hierzu verwendete Schnittstellenzeiger ist immer gültig (ungleich Null) und vom Typ I_TcMessage. Insbesondere ist der entsprechende Fehler im PLC-OnlineView sofort als Klartext zu sehen.
stInfo	ST_CoE_DiagHistory_Info [▶ 13]	Liefert Informationen vom ausgelesenen EtherCAT Slave Diag History Object. Dies beinhaltet u.a. den verwendeten Buffer Mode und Informationen zu den verfügbaren Meldungen. Dieser Ausgang wird nur nach erfolgreichem Lesen aktualisiert.

Voraussetzungen

TwinCAT Version	Hardware	Einzubindende Bibliotheken
TwinCAT 3.1 >= Build 4024.57	x86, x64, ARM	Tc3_EtherCATDiag

4 Datentypen

4.1 ST_CoE_DiagHistory_Entry

Syntax

```
TYPE ST_CoE_DiagHistory_Entry :
STRUCT
    eMsgType      :  E_CoE_0x10F3_DiagHistory_MsgType;
    bAckStatus    :  BOOL;
    stTimeStamp   :  TIMESTRUCT;
    {attribute 'displaymode':='hex'}
    nDiagCode     :  UDINT;
    eDiagCodeType :  E_CoE_0x10F3_DiagHistory_DiagCodeType;
    {attribute 'displaymode':='hex'}
    nTextId       :  UINT;
    nModuleNo     :  UINT;
    sMsgText      :  STRING(DiagHistory_Parameters.cMsgStringLength);
END_STRUCT
END_TYPE
```

Parameter

Name	Typ	Beschreibung
eMsgType	E_CoE_0x10F3_DiagHistory_MsgType [▶ 16]	Art der Meldung/Diagnose Dies kann mittels der TO_STRING() Methode in eine Zeichenkette umgewandelt werden.
bAckStatus	BOOL	Status der bestätigten Meldung
stTimeStamp	TIMESTRUCT	Zeitstempel
nDiagCode	UDINT	Diagnosecode - numerischer Wert
eDiagCodeType	E_CoE_0x10F3_DiagHistory_DiagCodeType [▶ 15]	Typisierung des Diagnosecode Dies kann mittels der TO_STRING() Methode in eine Zeichenkette umgewandelt werden.
nTextId	UINT	Text-ID
nModuleNo	UINT	Diagnosecode-Modulnummer
sMsgText	STRING	Meldungstext

4.2 ST_CoE_DiagHistory_Info

Die Struktur liefert Informationen zu den Einstellungen im EtherCAT Slave hinsichtlich der EtherCAT Diagnose.

Syntax

```
TYPE ST_CoE_DiagHistory_Info :
STRUCT
    nMaxNoOfMsgs            : UINT;
    eBufferMode             : E_CoE_0x10F3_DiagHistory_BufferMode;
    bNewMsgsAvail           : BOOL;
    bEmergSendingEnabled    : BOOL;
    bInfoMsgsDisabled       : BOOL;
    bWarnMsgsDisabled       : BOOL;
    bErrMsgsDisabled        : BOOL;
    bOldMsgsOverwritten     : BOOL;
    bNewMsgsDiscarded       : BOOL;
END_STRUCT
END_TYPE
```

Parameter

Name	Typ	Beschreibung
nMaxNoOfMsgs	UINT	Maximale Anzahl von Meldungen im Puffer des EtherCAT Slave
eBufferMode	E_CoE_0x10F3_DiagHistory_BufferMode [▶ 15]	Überschreib- oder Bestätigungsmodus des EtherCAT Slave
bNewMsgsAvail	BOOL	Neue Nachrichten seit dem letzten Lesen verfügbar.
bEmergSendingEnabled	BOOL	Notfallnachrichten aktiviert. Neue Diagnosenachrichten müssen als Notfallnachricht gesendet werden.
bInfoMsgsDisabled	BOOL	Informationsmeldungen deaktiviert.
bWarnMsgsDisabled	BOOL	Warnmeldungen deaktiviert.
bErrMsgsDisabled	BOOL	Fehlermeldungen deaktiviert.
bOldMsgsOverwritten	BOOL	Alte Nachrichten im Puffer wurden durch neue Nachrichten überschrieben (Overwrite Mode).
bNewMsgsDiscarded	BOOL	Neue Nachrichten wurden verworfen, weil der Puffer mit unbestätigten Nachrichten gefüllt ist (Acknowledge-Modus).

4.3 ST_CoE_DiagHistory_Options

Syntax

```
TYPE ST_CoE_DiagHistory_Options :
STRUCT
    nLangId                                 : DINT := 1033;
    eBufferMode                             : E_CoE_0x10F3_DiagHistory_BufferMode :=
E_CoE_0x10F3_DiagHistory_BufferMode.OverwriteMode;
    bEnableEmergSending                     : BOOL := TRUE;
    bDisableInfoMsgs                        : BOOL := FALSE;
    bDisableWarnMsgs                        : BOOL := FALSE;
    bDisableErrMsgs                         : BOOL := FALSE;
    bDisableAutoRecoverAfterConnectionErr   : BOOL := FALSE;
    bExcludeTextIdFromMsgText               : BOOL := FALSE;
END_STRUCT
END_TYPE
```

Parameter

Name	Typ	Beschreibung
nLangId	DINT	Optional: Sprach-ID, 1033 = Englisch (US), 1031 = Deutsch usw.
eBufferMode	E_CoE_0x10F3_DiagHistory_BufferMode [▶ 15]	Überschreib- oder Bestätigungsmodus
bEnableEmergSending	BOOL	Das Senden von Notfallmeldungen wird deaktiviert. Neue Diagnosenachrichten werden nicht als Notfallnachricht gesendet.
bDisableInfoMsgs	BOOL	Infomeldungen werden deaktiviert.
bDisableWarnMsgs	BOOL	Warnmeldungen werden deaktiviert.
bDisableErrMsgs	BOOL	Fehlermeldungen werden deaktiviert.
bDisableAutoRecoverAfterConnectionErr	BOOL	Die automatische Wiederherstellung der Verbindung zur EtherCAT Diagnose Historie wird hiermit deaktiviert. Per Default wird die Verbindung wieder hergestellt nach einem Verbindungsfehler, sofern der Eingang bEnable weiterhin TRUE ist.
bExcludeTextIdFromMsgText	BOOL	Durch Setzen dieser Option wird die Text ID aus dem Nachrichtentext entfernt.

4.4 E_CoE_0x10F3_DiagHistory_BufferMode

Der Buffer Mode legt fest, wie sich der EtherCAT Slave verhält hinsichtlich seines Fehlerspeichers. Die Meldungen werden in Form eines Ringspeichers vorgehalten.

```
TYPE E_CoE_0x10F3_DiagHistory_BufferMode :
(
UNKNOWN,
OverwriteMode,
AcknowledgeMode
) USINT;
END_TYPE
```

Wert	Beschreibung
OverwriteMode	Im Overwrite Mode werden, sofern der Ringspeicher voll ist, immer die ältesten Meldungen verworfen.
AcknowledgeMode	Im Acknowledge Mode werden, sofern der Ringspeicher voll ist, die ältesten bestätigten Meldungen verworfen. Sind nur unbestätigte Meldungen in der Liste, verbleiben diese und die neuesten einkommenden Meldungen werden verworfen. Werden Meldungen mit dem Bausteineingang bAck bestätigt, so können diese demnach von neuen Meldungen überschrieben.

4.5 E_CoE_0x10F3_DiagHistory_DiagCodeType

Die Enumeration spezifiziert den Typ des ausgegebenen DiagCode Wertes. Die Typen sind durch die ETG Norm definiert.

```
TYPE E_CoE_0x10F3_DiagHistory_DiagCodeType :
(
    UNKNOWN,
    NotUsed,
    ManufacturerSpecific,
    EmergencyErrorCode,
    Reserved,
    ModuleRelated,
    DeviceRelated,
    ProfileSpecific
) USINT;
END_TYPE
```

Wert	Beschreibung
UNKNOWN	Unbekannter Typ von Meldung
NotUsed	Der DiagCode wird aktuell nach Norm nicht verwendet.
ManufacturerSpecific	Der DiadCode ist spezifisch und vom Gerätehersteller definiert.
EmergencyErrorCode	Bei dem DiagCode handelt es sich um einen Emergency Code.
Reserved	Der DiagCode ist reserviert für zukünftige Verwendung. Der Typ ist aktuell noch nicht definiert.
ModuleRelated	Der DiagCode ist modulbezogen.
DeviceRelated	Der DiagCode ist gerätebezogen.
ProfileSpecific	Der DiagCode ist spezifisch für das Profil.

4.6 E_CoE_0x10F3_DiagHistory_MsgType

```
TYPE E_CoE_0x10F3_DiagHistory_MsgType :
(
    UNKNOWN,
    Info,
    Warning,
    Error,
    RESERVED_Value_x
) USINT;
END_TYPE
```

Wert	Beschreibung
UNKNOWN	Unbekannter Typ von Meldung
Info	Es handelt sich um eine Informationsmeldung.
Warning	Es handelt sich um eine Warnmeldung.
Error	Es handelt sich um eine Fehlermeldung.
RESERVED_Value_x	Weitere Typen von Meldungen sind aktuell nicht definiert. Die Werte sind für die Zukunft reserviert.

5 Globale Parameter

5.1 DiagHistory_Parameters

Diese Parameter sind zur Laufzeit konstant, können jedoch über den Bibliotheksmanager projektspezifisch verändert werden und werden mit den Projekteigenschaften abgespeichert. Sie ermöglichen durch Anpassung längere Messagetexte oder mehr Argumente innerhalb der Messagetexte. Aufgrund der großen, teils internen, Arrays wächst ggf. die Größe der Bausteininstanzen und auch die Programmgröße durch Anpassung der Parameter stark an.

Syntax

```
VAR_GLOBAL CONSTANT
    // FB-internal read buffer - Max.  parameters/arguments buffer size per message in bytes.
    // Reduce to optimize memory resources. Increase if needed.
    cMsgParsBufferSize    : UINT(0..1000) := 128;

    // FB-internal read buffer - Max number of messages in diag history buffer.
    // (ETG.1020 Norm states 250 max. possible messages. Not all Slaves hold this many messages).
    // Reduce to optimize memory resources. Increase if needed.
    cMaxMsgsBuffer        : UINT(0..250)  := 250;

    // FB-internal read buffer - Max length of message string. Reduce to optimize memory resources.
    // Increase if needed. Increase if needed.
    cMsgStringLength      : UINT(0..255) := 200;
END_VAR
```

6 Globale Konstanten

6.1 Global_Version

Alle Bibliotheken haben eine bestimmte Version. Diese Version ist u. a. im SPS-Bibliotheks-Repository zu sehen. Eine globale Konstante enthält die Information über die Bibliotheksversion:

```
VAR_GLOBAL CONSTANT
   stLibVersion_Tc3_EtherCATDiag : ST_LibVersion;
END_VAR
```

Name	Typ	Beschreibung
stLibVersion_Tc3_EtherCATDiag	ST_LibVersion	Versionsinformation der Tc3_EtherCATDiag-Bibliothek

Um zu sehen, ob die Version, die Sie haben, auch die Version ist, die Sie brauchen, benutzen Sie die Funktion F_CmpLibVersion (definiert in der SPS-Bibliothek Tc2_System).

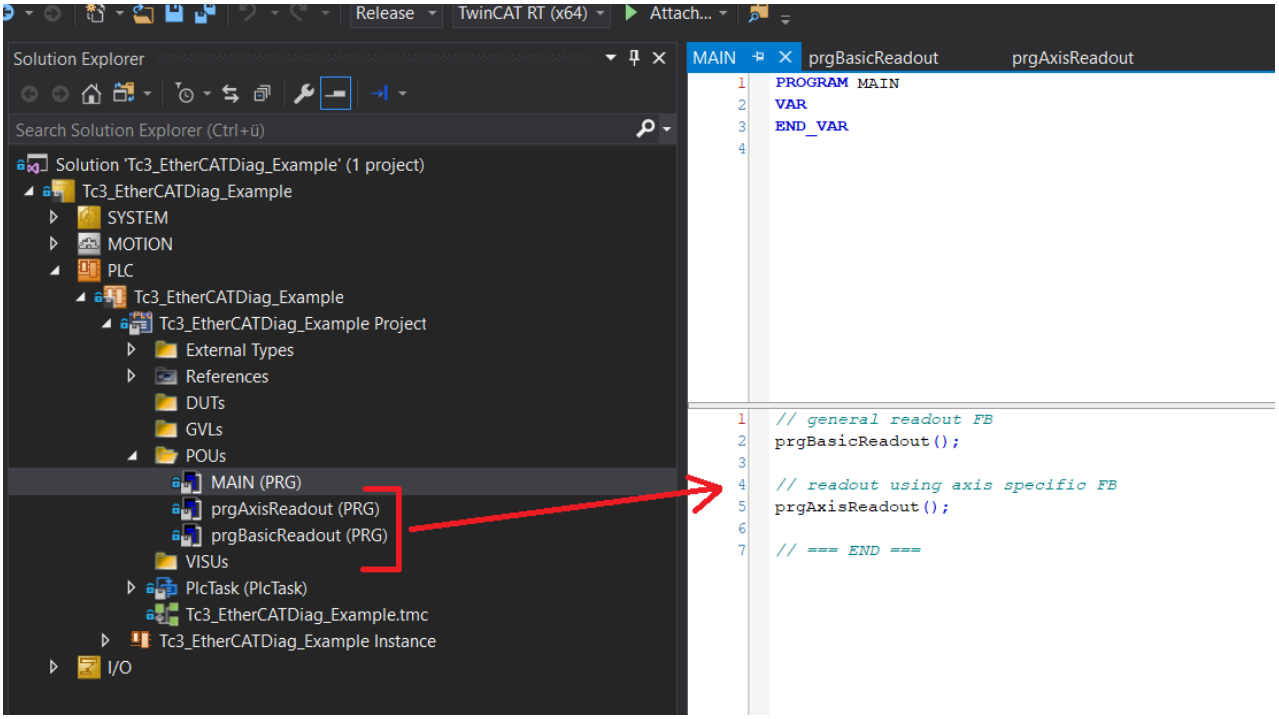
7 Beispiele

Dokumente hierzu

Tc3_EtherCATDiag_Example (Resources/zip/16600107531.zip)

7.1 Tc3_EtherCATDiag Example

Das Beispielprojekt enthält zwei Programm-POUs, die jeweils einen Beispielaufruf des `FB_CoE_DiagHistory_Read` und des `FB_CoE_DiagHistory_ReadAxis` enthalten:



Beispielaufruf FB_CoE_DiagHistory_Read

Das Beispiel enthält einen simplen Aufruf des Basis-FBs. Die Adresse des Slaves wird über die Eingänge sNetId und nSlaveAddr verlinkt. Am Eingang bEnable kann der Baustein entweder einmalig (bReadOnce) oder dauerhaft (bReadContinous) angesteuert werden. Mit dem Eingang bAck können Meldungen quittiert werden. Sonstige Ansteuerungen oder Konfigurationen können mittels stOptions angesteuert werden. Die Meldungen des Slaves werden in dem Array aMsgArray gespeichert. Die Signale bReadOnce und bAck werden zyklisch FALSE geschrieben.

prgBasicReadout | prgAxisReadout

```
PROGRAM prgBasicReadout
VAR
    bReadOnce                   : BOOL;
    bReadContinuous             : BOOL;
    bAck                        : BOOL;
    stOptions                   : ST_CoE_DiagHistory_Options;

    fbCoEDiagHist               : FB_CoE_DiagHistory_Read;
    IstAmsAdr           AT %I*  : AMSADDR;
    aMsgArray                   : ARRAY[1..100] OF ST_CoE_DiagHistory_Entry;
END_VAR
```

```
fbCoEDiagHist(
    sNetId                  :=  F_CreateAmsNetId(IstAmsAdr.netId),
    nSlaveAddr              :=  IstAmsAdr.port,
    bEnable                 :=  bReadOnce
                                OR bReadContinuous,
    bAck                    :=  bAck,
    tTimeout                :=  ,
    tReadInterval           :=  ,
    stOptions               :=  stOptions,
    aDiagHistory            :=  aMsgArray,
    bBusy                   =>  ,
    bValid                  =>  ,
    bError                  =>  ,
    hrErrorCode             =>  ,
    ipErrorMessage          =>  ,
    stInfo                  =>  );

bReadOnce   :=  FALSE;
bAck        :=  FALSE;

// === END ===
```

Beispielaufruf FB_CoE_DiagHistory_ReadAxis

Das Beispiel enthält einen simplen Aufruf des Achs-FBs. Die Adresse des Slaves wird mittels des refAxis ermittelt. Am Eingang bEnable kann der Baustein entweder einmalig (bReadOnce) oder dauerhaft (bReadContinous) angesteuert werden. Mit dem Eingang bAck können Meldungen quittiert werden. Sonstige Ansteuerungen oder Konfigurationen können mittels stOptions angesteuert werden. Die Meldungen des Slaves werden in dem Array aMsgArray gespeichert. Die Signale bReadOnce und bAck werden zyklisch FALSE geschrieben.

MAIN prgBasicReadout prgAxisReadout

```
PROGRAM prgAxisReadout
VAR
    bReadOnce                   : BOOL;
    bReadContinuous             : BOOL;
    bAck                        : BOOL;
    stOptions                   : ST_CoE_DiagHistory_Options;

    fbCoEDiagHist_Axis          : FB_CoE_DiagHistory_ReadAxis;
    refAxis                     : AXIS_REF;
    aMsgArray                   : ARRAY[1..100] OF ST_CoE_DiagHistory_Entry;
END_VAR
```

```
fbCoEDiagHist_Axis(
    refAxis                 :=  refAxis,
    sNetId                  :=  ,
    nSlaveAddr              :=  ,
    bEnable                 :=  bReadOnce
                                OR bReadContinuous,
    bAck                    :=  bAck,
    tTimeout                :=  ,
    tReadInterval           :=  ,
    stOptions               :=  stOptions,
    aDiagHistory            :=  aMsgArray,
    bBusy                   =>  ,
    bValid                  =>  ,
    bError                  =>  ,
    hrErrorCode             =>  ,
    ipErrorMessage          =>  ,
    stInfo                  =>  );

bReadOnce   :=  FALSE;
bAck        :=  FALSE;

// === END ===
```

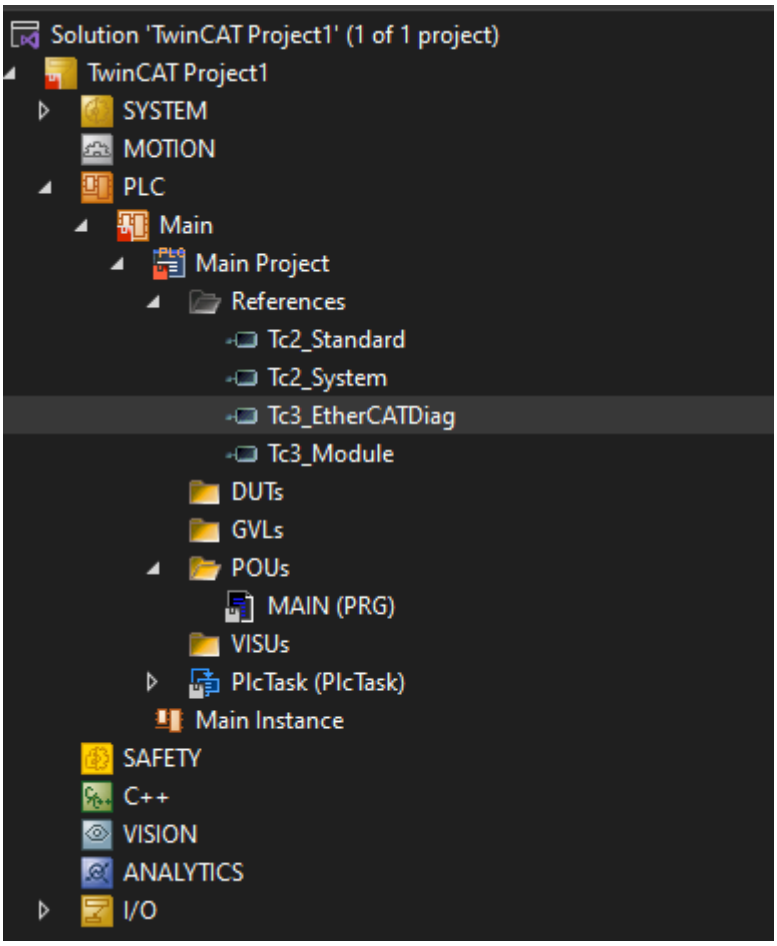
Download: https://infosys.beckhoff.com/content/1031/tcplclib_tc3_ethercatdiag/Resources/16600107531.zip

7.2 Tutorial

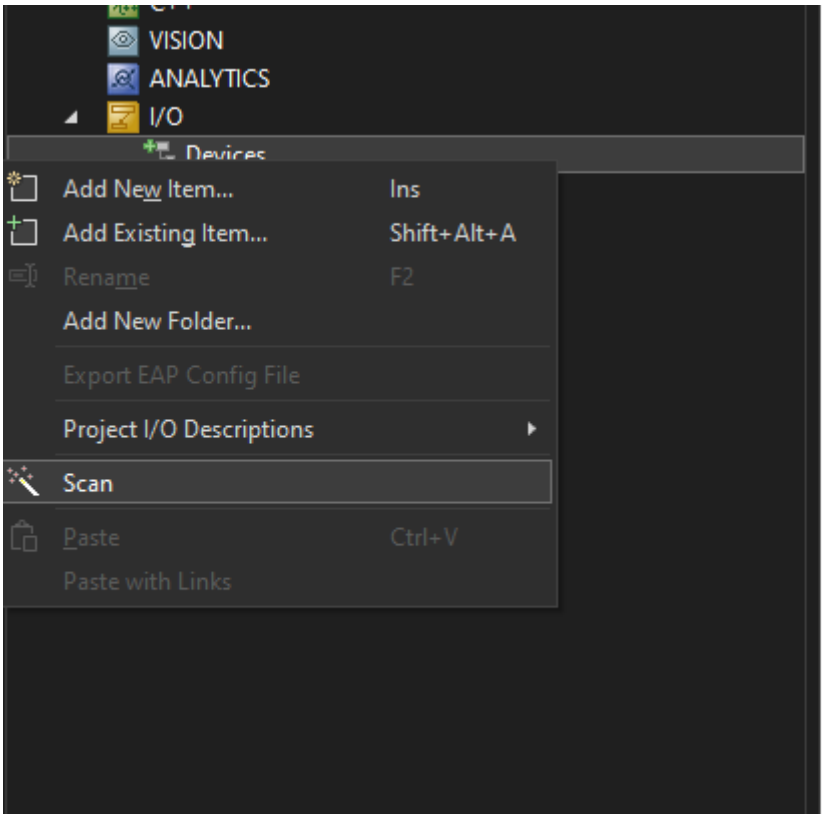
FB_CoE_DiagHistory_Read

In dem folgenden Tutorial wird eine einfache Implementierung des FB_CoE_DiagHistory_Read dokumentiert.

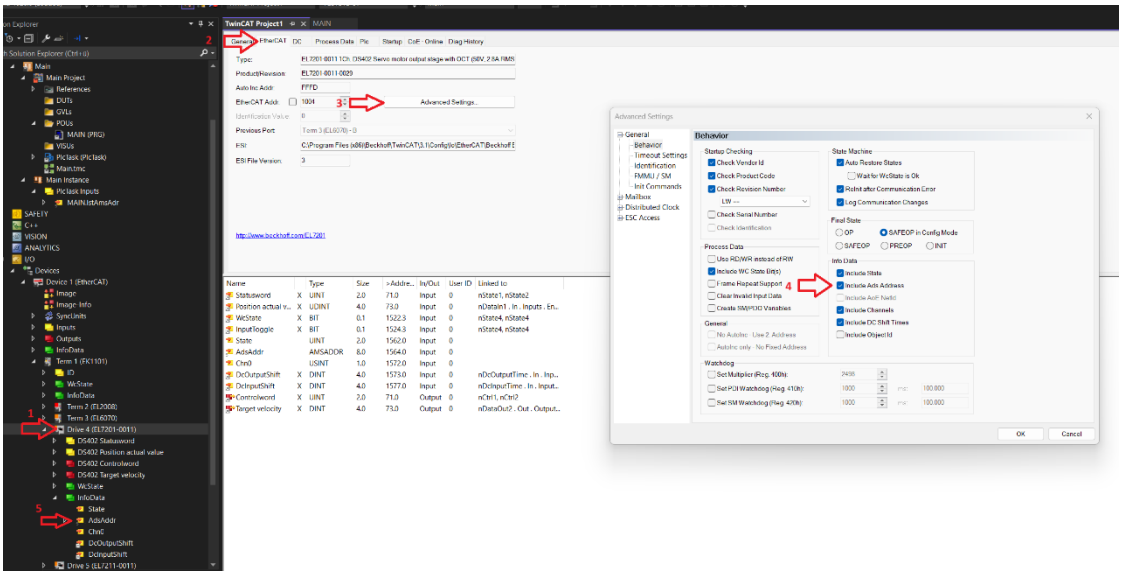
1. Erstellen Sie eine neue TwinCAT Solution und binden Sie die Bibliotheken Tc3_EtherCATDiag ein.



⇨ Die verbundene Hardware kann per „Scan“ eingelesen werden.



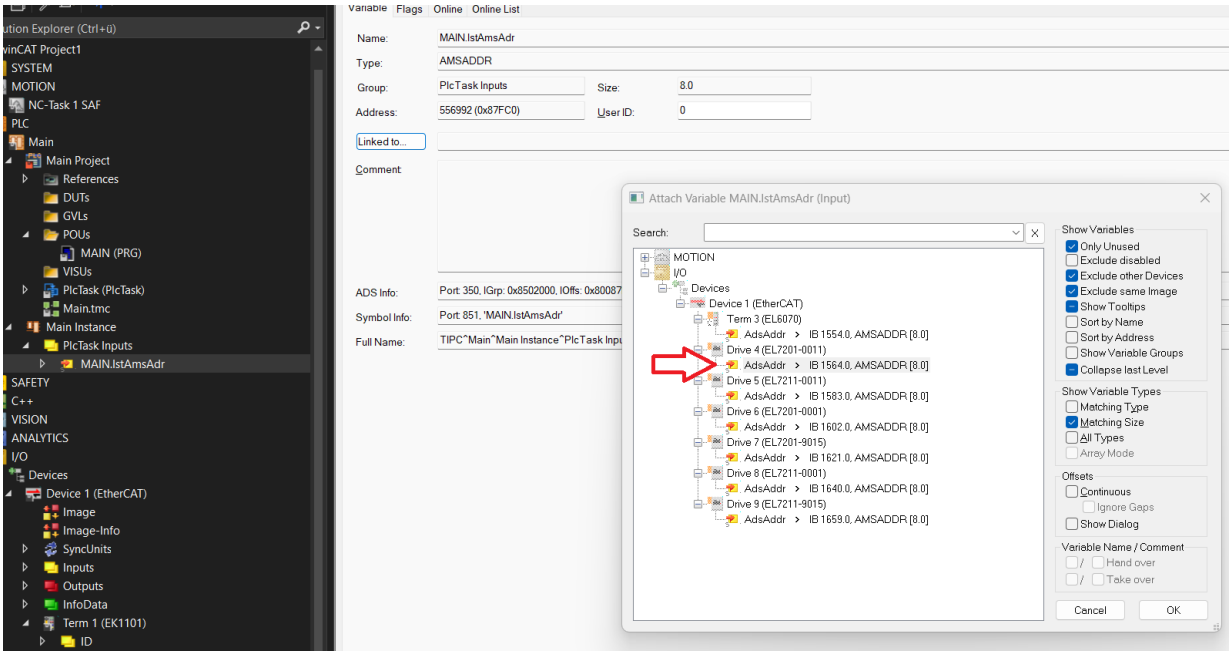
2. Anschließend können Sie die ADS-Adresse des gewünschten Slaves in den „Advanced EtherCAT Settings“ sichtbar/verlinkbar machen.



3. Nun können Sie die folgenden Variablen deklarieren.

```
MAIN*
1   PROGRAM MAIN
2   VAR
3       bReadOnce                   : BOOL;
4       bReadContinuous             : BOOL;
5       bAck                        : BOOL;
6       stOptions                   : ST_CoE_DiagHistory_Options;
7
8       fbCoEDiagHist               : FB_CoE_DiagHistory_Read;
9       IstAmsAdr           AT %I*  : AMSADDR;
10      aMsgArray                   : ARRAY[1..100] OF ST_CoE_DiagHistory_Entry;
11  END_VAR
12
```

⇨ IstAmsAdr wird mit der AMS-Adresse des gewünschten Slaves verlinkt.



⇨ Der Aufruf des FBs erfolgt wie im Beispielprojekt.

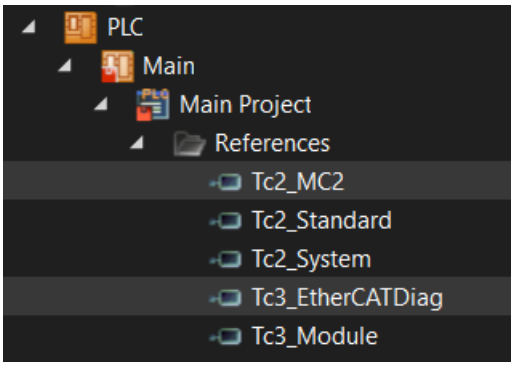
```
fbCoEDiagHist(
    sNetId                  :=  F_CreateAmsNetId(IstAmsAdr.netId),
    nSlaveAddr              :=  IstAmsAdr.port,
    bEnable                 :=  bReadOnce
                                OR bReadContinuous,
    bAck                    :=  bAck,
    tTimeout                :=  ,
    tReadInterval           :=  ,
    stOptions               :=  stOptions,
    aDiagHistory            :=  aMsgArray,
    bBusy                   =>  ,
    bValid                  =>  ,
    bError                  =>  ,
    hrErrorCode             =>  ,
    ipErrorMessage          =>  ,
    stInfo                  =>  );

bReadOnce    :=  FALSE;
bAck         :=  FALSE;
```

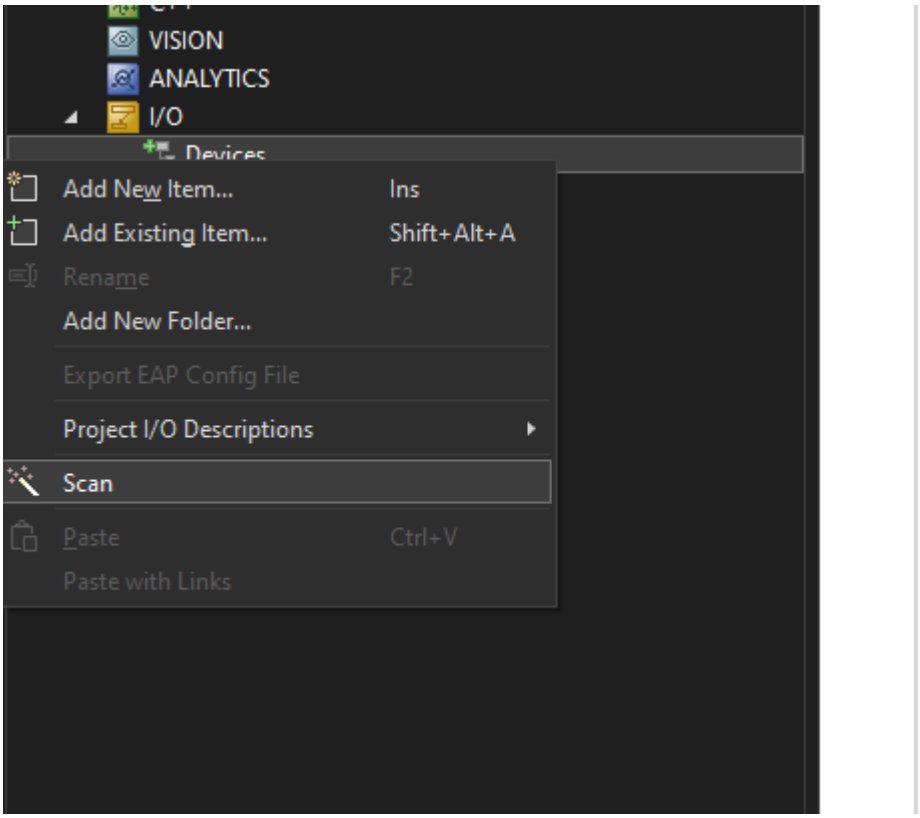
FB_CoE_DiagHistory_ReadAxis

In dem folgenden Tutorial wird eine einfache Implementierung des FB_CoE_DiagHistory_ReadAxis dokumentiert.

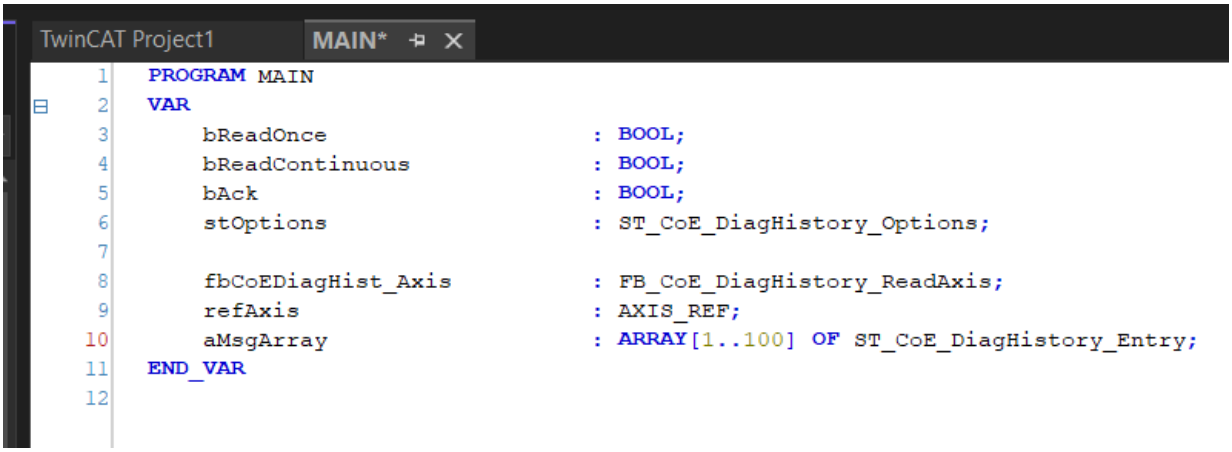
1. Erstellen Sie eine neue TwinCAT Solution und binden Sie die Bibliotheken Tc3_EtherCATDiag und Tc2_MC2 ein.



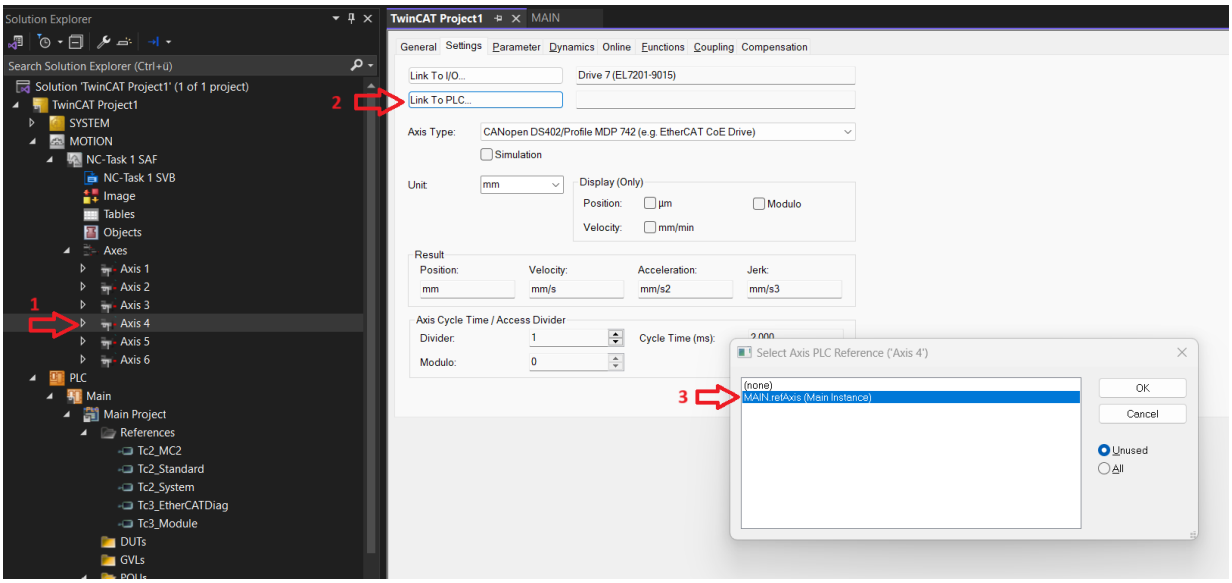
2. Sie können die verbundene Hardware per „Scan“ einlesen. Motion-Achsen können Sie in diesem Schritt automatisch anlegen.



3. Sie können folgende Variablen deklarieren.



4. Verlinken Sie das refAxis mit der gewünschten Achse.



⇨ Der Aufruf des FBs erfolgt wie im Beispielprojekt:

```
1  fbCoEDiagHist_Axis(
2      refAxis                 :=  refAxis,
3      sNetId                  :=  ,
4      nSlaveAddr              :=  ,
5      bEnable                 :=  bReadOnce
6                                  OR bReadContinuous,
7      bAck                    :=  bAck,
8      tTimeout                :=  ,
9      tReadInterval           :=  ,
10     stOptions               :=  stOptions,
11     aDiagHistory            :=  aMsgArray,
12     bBusy                   =>  ,
13     bValid                  =>  ,
14     bError                  =>  ,
15     hrErrorCode             =>  ,
16     ipErrorMessage          =>  ,
17     stInfo                  =>  );
18
19 bReadOnce    :=  FALSE;
20 bAck         :=  FALSE;
21
```

Mehr Informationen:
www.beckhoff.com/te1000

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Deutschland
Telefon: +49 5246 9630
info@beckhoff.com
www.beckhoff.com