acontis technologies GmbH

SOFTWARE

EC-Master

Feature Pack Superset-ENI

Version 3.2

Edition: August 1, 2025

Contents

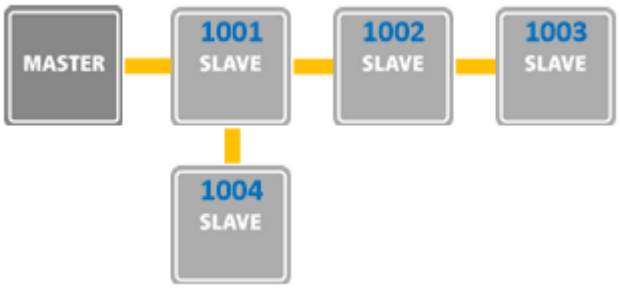
1 Introduction

The Superset ENI functionality allows the user to adapt the configuration without generating a new ENI file. The base ENI file contains a superset of all the possible connected slaves. According to the use case the application can exclude slaves from this superset.

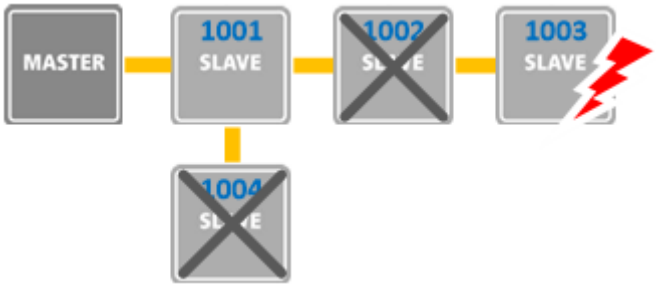
1.1 Optional Slaves

A possible use case of Superset ENI is that a configuration contains several optional slaves that can be excluded depending on their presence.

The following figure shows a simple network configuration:



If Slave 1002 and Slave 1004 are not present, the bus mismatches:



The following source code removes the missing Slaves from the configuration after loading the ENI file, so the resulting topology validates:

```
dwRes = emInitMaster(dwInstanceId, &oInitMasterParms);

/* load configuration from ENI */
dwRes = emConfigLoad(dwInstanceId, eCnfType_Filename, (EC_T_BYTE*)"eni.xml", (EC_T_
↪DWORD)OsStrlen("eni.xml"));

/* exclude slaves from configuration */
dwRes = emConfigExcludeSlave(dwInstanceId, 1002);
dwRes = emConfigExcludeSlave(dwInstanceId, 1004);

/* apply modified configuration */
dwRes = emConfigApply(dwInstanceId);
```

Resulting topology:

1.2 Alternative Slaves

Another use case of Superset ENI would be that a configuration contains several different alternatives for a slave at a certain position.

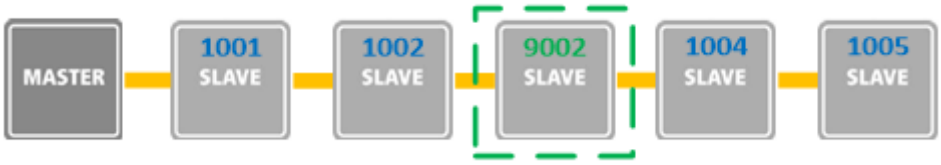
The following network includes one slave which may be of different types:

To realize this, an ENI containing all possible slave definitions is needed, e.g. the two Slaves 9001 and 9002:

The following source code first removes all alternatives from the configuration after loading the ENI file and then include one of the alternative slaves so that the resulting topology is valid:

```
dwRes = emInitMaster(dwInstanceId, &oInitMasterParms);

/* load configuration from ENI */
dwRes = emConfigLoad(dwInstanceId, eCnfType_Filename, (EC_T_BYTE*)"eni.xml", (EC_T_
↪DWORD)OsStrlen("eni.xml"));

/* exclude all alternative slaves from configuration */
dwRes = emConfigExcludeSlave(dwInstanceId, 9001);
dwRes = emConfigExcludeSlave(dwInstanceId, 9002);

/* ... */

/* select alternative slave from configuration */
dwRes = emConfigIncludeSlave(dwInstanceId, 9002);

/* apply modified configuration */
dwRes = emConfigApply(dwInstanceId);
```

Resulting topology:


MASTER
1001
SLAVE
1002
SLAVE
9002
SLAVE
1004
SLAVE
1005
SLAVE

2 Application programming interface, reference

2.1 emConfigLoad

static EC_T_DWORD **ecatConfigLoad** (
    EC_T_CNF_TYPE eCnfType,
    EC_T_PBYTE pbyCnfData,
    EC_T_DWORD dwCnfDataLen
)

EC_T_DWORD **emConfigLoad** (
    EC_T_DWORD dwInstanceID,
    EC_T_CNF_TYPE eCnfType,
    EC_T_PBYTE pbyCnfData,
    EC_T_DWORD dwCnfDataLen
)

Load the network configuration.

In combination with emConfigApply, this function replaces emConfigureNetwork and must be called after the initialization. Among others the EtherCAT topology defined in the given XML configuration file will be stored internally.

Note: A client must not be registered prior to calling this function. Existing client registrations will be dropped.

Parameters

- **dwInstanceID** – [in] Instance ID (Multiple EtherCAT Network Support)
- **eCnfType** – [in] Type of configuration data provided
- **pbyCnfData** – [in] Configuration data
- **dwCnfDataLen** – [in] Length of configuration data in byte

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if master isn't initialized
- EC_E_INVALIDPARM if dwInstanceID is out of range, the input pointer is EC_NULL or contains EC_NULL pointer
- EC_E_NOMEMORY if some memory cannot be allocated
- EC_E_ADS_IS_RUNNING if ADS server is running

enum **EC_T_CNF_TYPE**

Values:

enumerator **eCnfType_Unknown**

enumerator **eCnfType_Filename**

pbyCnfData: ENI filename to read

enumerator **eCnfType_Data**
pbyCnfData: ENI data

enumerator **eCnfType_Datadiag**
pbyCnfData: ENI data for diagnosis

enumerator **eCnfType_GenPreopENI**
Generate ENI based on bus-scan result to get into PREOP state

enumerator **eCnfType_GenPreopENIWithCRC**
Same as eCnfType_GenPreopENI with CRC protection

enumerator **eCnfType_GenOpENI**
Generate ENI based on bus-scan result to get into OP state. The default PDO mapping read from the slaves is activated. See ETG2010 "SII Specification", Table 14 "Structure Category TXPDO and RXPDO for each PDO"

enumerator **eCnfType_None**
Reset configuration

enumerator **eCnfType_ConfigData**
pbyCnfData: Binary structured configuration

enumerator **eCnfType_GenOpENINoStrings**
Generate ENI based on bus-scan result to get into OP state , does not read strings from EEPROM

enumerator **eCnfType_FileByApp**
File access provided by user application, See EC_T_CNF_FILEBYAPP_DESC

enumerator **eCnfType_GenEBI**
Generate EBI based on bus-scan result

2.2 emConfigExcludeSlave

static EC_T_DWORD **ecatConfigExcludeSlave** (EC_T_WORD wStationAddress)

EC_T_DWORD **emConfigExcludeSlave** (
EC_T_DWORD dwInstanceID,
EC_T_WORD wStationAddress
)

Exclude a slave from the network configuration.

It has to be called after emConfigLoad and prior to calling emConfigApply.

Parameters

- **dwInstanceID** – [in] Instance ID (Multiple EtherCAT Network Support)
- **wStationAddress** – [in] Station address of the slave to be excluded. A value of 0 excludes all slaves.

Returns
EC_E_NOERROR or error code

2.3 emConfigIncludeSlave

static EC_T_DWORD **ecatConfigIncludeSlave** (EC_T_WORD wStationAddress)

EC_T_DWORD **emConfigIncludeSlave** (
    EC_T_DWORD dwInstanceID,
    EC_T_WORD wStationAddress
)

Include a slave in the network configuration.

Slaves that were previously excluded with emConfigSlaveExclude can be added again. It has to be called after emConfigLoad and prior to calling emConfigApply.

Parameters

- **dwInstanceID** – [in] Instance ID (Multiple EtherCAT Network Support)
- **wStationAddress** – [in] Station address of the slave to be included. A value of 0 includes all slaves.

Returns

EC_E_NOERROR or error code

2.4 emConfigSetPreviousPort

static EC_T_DWORD **ecatConfigSetPreviousPort** (
    EC_T_WORD wStationAddress,
    EC_T_WORD wStationAddressPrev,
    EC_T_WORD wPortPrev
)

EC_T_DWORD **emConfigSetPreviousPort** (
    EC_T_DWORD dwInstanceID,
    EC_T_WORD wStationAddress,
    EC_T_WORD wStationAddressPrev,
    EC_T_WORD wPortPrev
)

Set previous port information of a slave.

It has to be called after emConfigLoad and prior to calling emConfigApply.

Parameters

- **dwInstanceID** – [in] Instance ID (Multiple EtherCAT Network Support)
- **wStationAddress** – [in] Station address of the slave
- **wStationAddressPrev** – [in] Previous slave station address
- **wPortPrev** – [in] Previous port

Returns

EC_E_NOERROR or error code

2.5 emConfigApply

static EC_T_DWORD **ecatConfigApply**(EC_T_VOID)

EC_T_DWORD **emConfigApply**(EC_T_DWORD dwInstanceID)

Apply the network configuration.

It has to be called after emConfigLoad.

Parameters

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

Returns

EC_E_NOERROR or error code