XTS TRANSPORT LAYER – a station based approach

BECKHOFF

XTS TRANSPORT LAYER

Daniel Hauer (HAUD)

BECKHOFF

1. Introduction

2. Requirements
   - XtsTransport (main control)
   - Xpu (XTS Processing Unit)
   - CaGroup (Collision Avoidance)
   - Mover (MC and CA)
   - Station (process handshake)

3. Design
   - use with any cyclic runtime
   - use with non cyclic software

4. License

1. Introduction

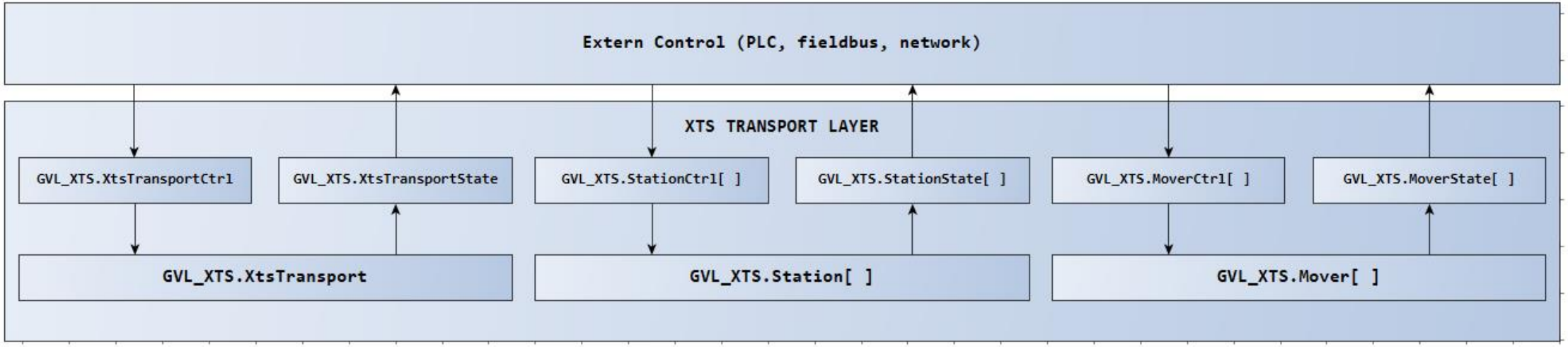
- This project collection is intended to convey the idea of a stand alone XTS transport layer to use in heterogen environments / applications.
- The main idea is that for every process a corresponding position on the xts exists.
- In order to reduce the amount of repetitive work when implementing a XTS into a machine, this project collection may help to put a transport layer in place
- A transport layer shall work a combination of discrete processes and continuous processes
- A transport layer shall have an interface for guiding a mover through a process station
- A transport layer shall have an interface to manipulate a mover
- A transport layer shall have an interface for setting-up or clearing the CollisionAvoidance Group

1. Introduction

- The XTS transport system enables a flexible product transport for various use cases.
- In combination with the Collision Avoidance library positioning of movers does not require extra monitoring of the axis
- Can be used for a station based approach, in which a station class is available for interaction with your process control
- Can be used for a mover based approach, your process control has a direct connection to every mover
- Can be used as a combination of station based and mover based approach
- The use of predefined datafields also enables control of XTS TRANSPORT LAYER via fieldbus or network.

1. Introduction

- designed for use with extern cyclic or non cyclic flow control (PLC, EtherCAT, any network)
  - Ctrl / State datafields for extern to access
- station based approach and individual manipulation of mover
- handshake in station with extern process flow (ST_STATION_CTRL / ST_STATION_STATE)
- individual cyclic mover interface with given set of movement functionalities (ST_MOVER_CTRL / ST_MOVER_STATE)



2. Requirements

BECKHOFF

- XtsTransport
  - Access to CA group function blocks (interface pointer)
  - Access to Stations (interface pointer)
  - Access to Movers (interface pointer)
  - Commands for getting all members to defined state
  - Cyclic interface for access from extern control
    - Ctrl (write): command
    - State (read): response to command
      - information from Xpu
      - Information from CA Group

2. Requirements

BECKHOFF

- Xpu (XTS Processing Unit)
  - Check Init Parameter
  - Check Online Parameter
  - Get Module Info Data
  - Connect TcCOM Objects to instances from XTS_Utility.lib function blocks
  - Cyclic plausibility checks
    - Mover ID detection after init
  - Cyclic interface for access from main control
    - Ctrl (write): command
    - State (read): response to command
    - Info (read): details from cyclic checks

2. Requirements

BECKHOFF

- CaGroup
  - Access to group function blocks
  - Access to movers for group commands
  - Get Group Info Data
  - Implements interface pointer for use in
    - Xts Transport

2. Requirements

BECKHOFF

- Mover
  - Access to MC function blocks
  - Access to CA function blocks
  - Cyclic interface for access from extern control
    - Ctrl (write): command
    - Data (write): command parameter
    - State (read): response to command
  - Implements Interface pointer for access from:
    - TransportUnit
    - Station
    - CaGroup

2. Requirements

BECKHOFF

- Station
  - Handshake mover transport with extern control
  - Close observation of movements with feedback to extern control
  - Linked List for movers in queue for infeed into station
  - Access to Linked List of target station for outfeed of mover
  - Cyclic interface for access from extern control
    - Ctrl (write): command and parameter
    - State (read): response to command and information about mover and queue
  - requires interface pointer to MC functionblocks

3. Design

BECKHOFF

- Namespace GVL_XTS
  - **Station**
    - Handshake with Process for mover transport
  - **XtsTransport**
    - Main command interface to extern control
  - **Xpu**
    - Access to TcCOM Objects
    - Cyclic plausibility checks
  - **CaGroup**
    - Access to CA library
  - **Mover**
    - Access to MC and CA library

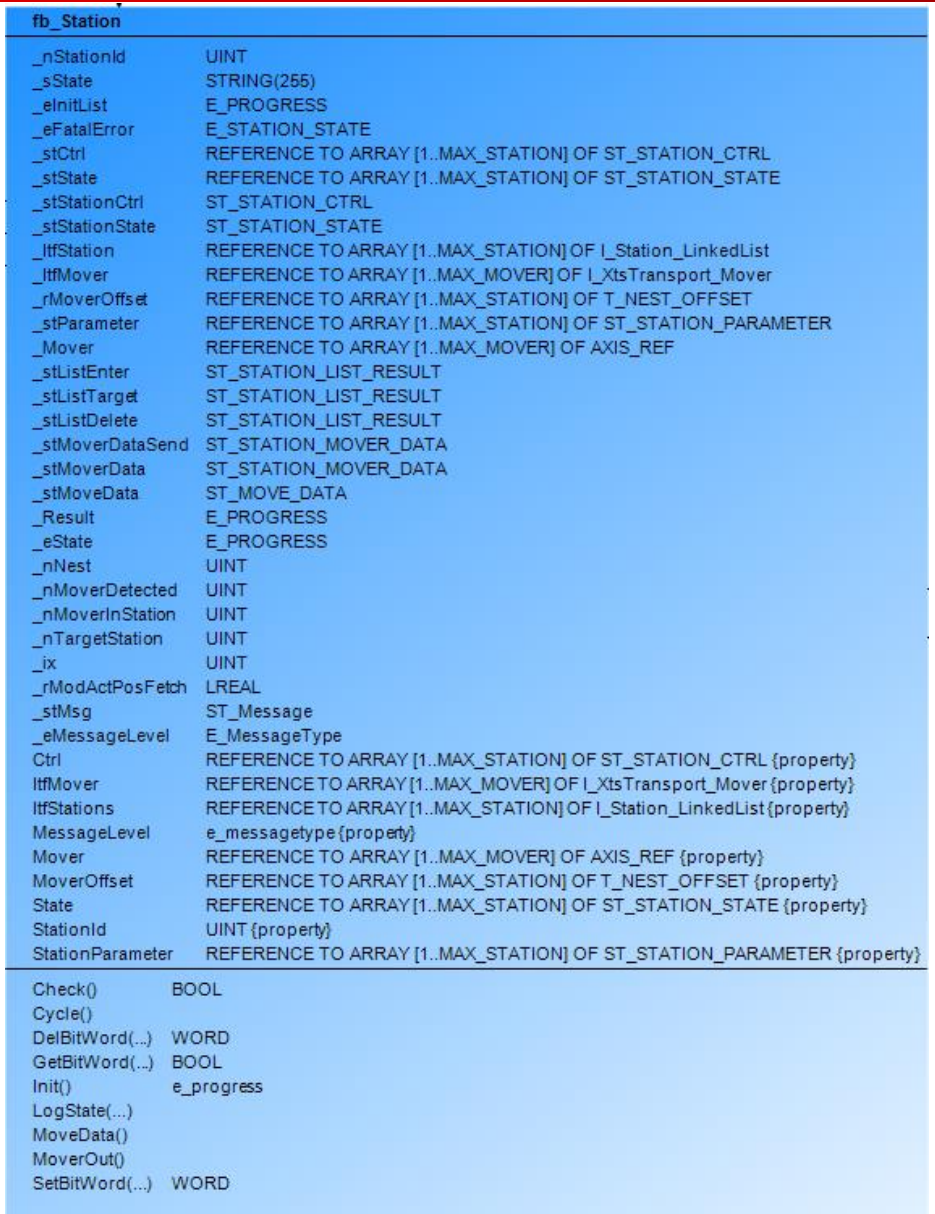
<<global>>
GVL_XTS

StationStart	ST_STATION_PARAMETER
Station	ARRAY [1..MAX_STATION] OF fb_Station
StationList	ARRAY [1..MAX_STATION] OF fb_Station_LinkedListCtrl
StationQueue	ARRAY [1..MAX_STATION] OF ARRAY [1..MAX_LIST_NODES] OF ST_STATION_MOVER_DATA
StationListItf	ARRAY [1..MAX_STATION] OF I_Station_LinkedList
StationCtrlItf	ARRAY [1..MAX_STATION] OF I_XtsTransport_Station
StationCtrl	ARRAY [1..MAX_STATION] OF ST_STATION_CTRL
StationState	ARRAY [1..MAX_STATION] OF ST_STATION_STATE
StationParameter	ARRAY [1..MAX_STATION] OF ST_STATION_PARAMETER
PositionOffset	ARRAY [1..MAX_STATION] OF T_NEST_OFFSET
XtsTransport	fb_TransportUnit
XtsTransportCtrl	ST_XTS_TRANSPORT_CTRL
XtsTransportState	ST_XTS_TRANSPORT_STATE
Xpu	fb_XpuCtrl
XpuCtrl	ST_XPU_CTRL
XpuState	ST_XPU_STATE
XpuInfo	ST_XPU_INFO
XpuModules	ARRAY [1..MAX_MODULE] OF Tc3_XTS_Utility.ST_InfoDataView
CaGroup	FB_CaGroup
CaGroupItf	I_XtsTransport_CaGroup
CaGroupRef	Tc3_McCoordinatedMotion.AXES_GROUP_REF
CaGroupInfo	ST_GROUP_INFO
Mover	ARRAY [1..MAX_MOVER] OF fb_MoverCtrl
MoverCtrl	ARRAY [1..MAX_MOVER] OF ST_MOVER_CTRL
MoverState	ARRAY [1..MAX_MOVER] OF ST_MOVER_STATE
MoverItf	ARRAY [1..MAX_MOVER] OF I_XtsTransport_Mover
LastPosition	ARRAY [1..MAX_MOVER] OF LREAL
LastGap	ARRAY [1..MAX_MOVER] OF LREAL
MoverInfo	ARRAY [1..MAX_MOVER] OF ST_MOVER_INFO
MoveData	ARRAY [1..MAX_MOVER] OF ST_MOVE_DATA
GearData	ARRAY [1..MAX_MOVER] OF ST_GEAR_DATA
AxisRefMover	ARRAY [1..MAX_MOVER] OF Tc2_MC2.AXIS_REF

3. Design

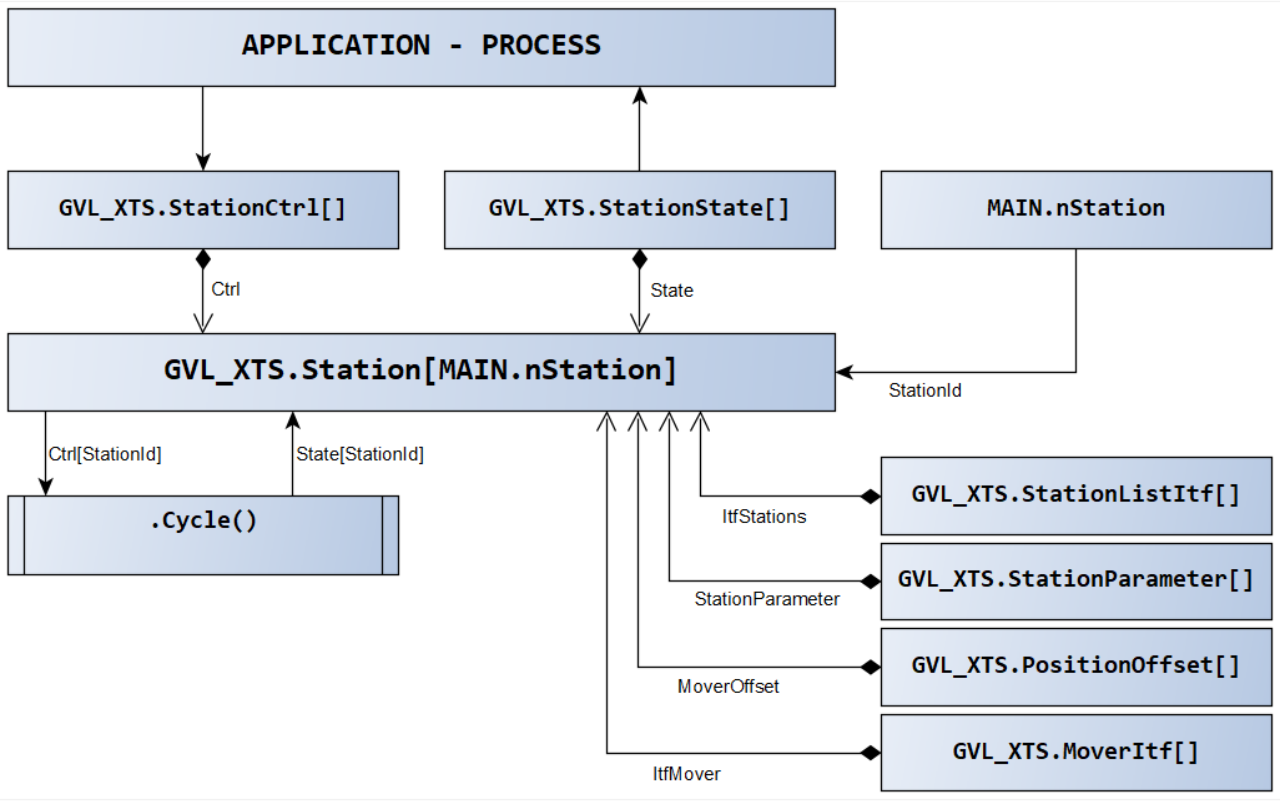
BECKHOFF

- GVL_XTS.Station
  - fb_Station[].Cycle
    - State machine for handshaking with extern control
      - Init (clears everything in station)
      - Enable
      - Mover Enter
      - Stop Position(s)
      - Mover Out
      - Empty
    - Control writes ticket for mover
      - MoverId
      - TargetStation
      - Mask
      - Offset



BECKHOFF

- GVL_XTS.Station
  - nStation index is passed as value from caller
  - Global datafields are passed as references (REF=) into fb_Station properties
    - Ctrl / State: handshakes
    - ItfStations: interface pointer to linked list methods for getting and setting of mover data
    - StationParameter: Coordinates and dynamic constraint of XtsStation
    - MoverOffset: correction values for every mover in every station with every nest (StopPos[])
    - ItfMover: interface pointer to CA movements



BECKHOFF

- GVL_XTS.Station (Planning requirements for use of fb_Station)
  - Put the Modulo turn anywhere, **BUT NOT** within PosWait, PosStop, ReleaseDistance of a station.
    The code does not support crossing the modulo turn within a station.
  - Since the project is designed for stations to send movers to a flexible target, with flexible nest positions, the control struct of a station you have to use, to forward those parameters together with the mover ID
    - **ST_STATION_CTRL.nMask:** commands the nest count and nest position of the mover in target station
    - **ST_STATION_CTRL.nTargetStation:** index of station in GVL_XTS.StationParameter[]
  - The Use of LinkedList methods (AddTail, GetHead) requires thought about when the mover is entered into the target station.
  - all coordinates are modulo values, from station to station only forward, within station limits backward movement by use of negative nest offset or use of **ST_MOVER_CTRL**.
  - IF move backwards required you have to make sure that there is room for it
    - Check PosStop
    - Each PosStop is relative to PosWait

BECKHOFF

- GVL_XTS.Station (Example)

- **ST_STATION_PARAMETER**: parallel Xts Stations for a process with common waiting position

  - Process uses GVL_XTS.Station[1] to GVL_XTS.Station[4]
    - Define PosWait(Queue position)
      - **[1].rPosWait := 100**
      - **[2].rPosWait := 100**
      - **[3].rPosWait := 100**
      - **[4].rPosWait := 100**
    - Define how many rPosStop(nests) the stations may have (configured count)
      - **[1].nConfiguredStopCount := 1 (default)**
      - **[2].nConfiguredStopCount := 1**
      - **[3].nConfiguredStopCount := 1**
      - **[4].nConfiguredStopCount := 1**

  - Process uses GVL_XTS.Station[1] to GVL_XTS.Station[4]
    - Define the process position(s) relative to rPosWait
      - **[1].rPosStop[1] := 100**
      - **[2].rPosStop[1] := 200**
      - **[3].rPosStop[1] := 300**
      - **[4].rPosStop[1] := 400**
    - The ReleaseDistance of the last station shall be shortest, all other stations follow accordingly.
      - **[1].rReleaseDistance := 40**
      - **[2].rReleaseDistance := 30**
      - **[3].rReleaseDistance := 20**
      - **[4].rReleaseDistance := 10**

BECKHOFF

- GVL_XTS.Station (Example)

- **using stations sparsely**:
  - In this case it is easiest to always handshake the stations and use the forwarding command if a station shall be skipped.
  - On Infeed state of mover, use: **E_STATION_CTRL. STATION_MOVER_SEND**.

- **deactivating stations**:
  - Make sure the queue is empty before deactivating, since the waiting mover will hold up all the others in case of required deactivation while movers are in the queue:
    - handshake mover with **E_STATION_CTRL.STATION_MOVER_SEND** to new target station if mover in queue cannot be processed
    - Handshake regular infeed if mover in queue can still be processed.
    - Do not send any new mover to the station in question
      - If queue of station is empty: **E_STATION_CTRL.STATION_DISABLE**
      - preceeding stations continue workflow with changed **ST_STATION_CTRL.nTargetStation**

BECKHOFF

- GVL_XTS.Station
  - Ctrl[nStation] : **ST_STATION_CTRL**
    - **eCmd (E_STATION_CTRL):**
      - enumeration for handshakes with State[nStation].eState **(E_STATION_STATE)**
    - **nMask (BYTE):**
      - bit mask to be used with multiple stop positions within a XtsStation.
        This mask tells the target station which StopPos[] (nest) has to be worked.
    - **nTargetStation (USINT):**
      - target to send mover to GVL_XTS.Station[nTargetStation].rPosWait
    - **rOffset (REAL):**
      - Optional offset for mover, used in target station in addition to static offset

BECKHOFF

- GVL_XTS.Station
  - State[nStation] : **ST_STATION_STATE**
    - **eState (E_STATION_STATE):**
      - Enumeration for active station state, Ctrl has to react to
    - **nMask (BYTE):**
      - Bitmask for active PosStop[] (nest)
    - **nMoverId (USINT):**
      - Active mover index in station
    - **rMoverModPos (LREAL):**
      - Modulo position of active mover
    - **nQueue (USINT):**
      - Count of movers, which were sent to XtsStation

3. Design

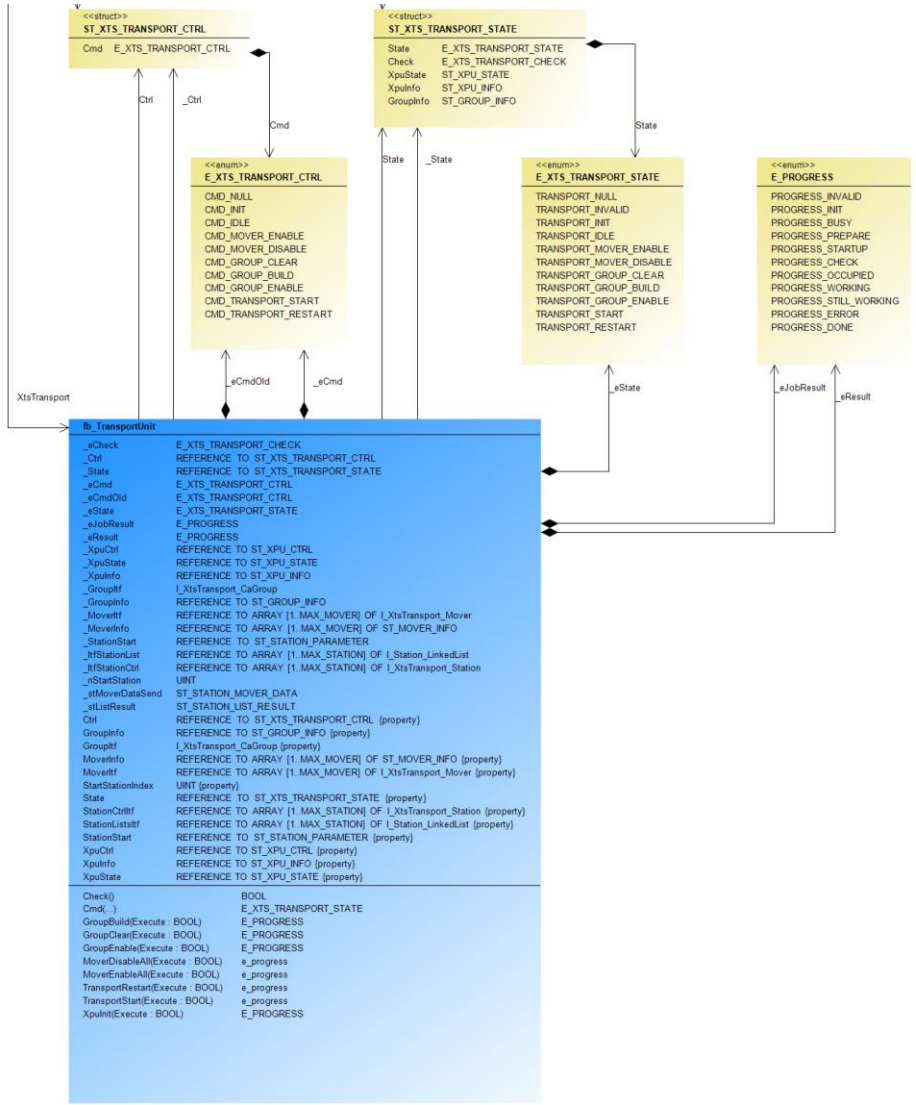
BECKHOFF

- GVL_XTS.StationParameter
  - **sText :**
    - Description only
  - **rPosWait :**
    - start of station, a sending station is using this value to send mover to
  - **rReleaseDistance :**
    - distance mover has to travel (from ActPos) in order for station to go back to mover detection
  - **rGap :**
    - Active gap on infeed and outfeed of station
  - **rVelo :**
    - Active velocity on infeed and outfeed of station
  - **rAccDec :**
    - Active dyn constraint
  - **rJerk :**
    - Active dyn constraint
  - **nConfiguredStopCount :**
    - Count of PosStop (nests) a mover may has to stop at in XtsStation
  - **rPosStop[] :**
    - Relative to rPosWait

3. Design

BECKHOFF

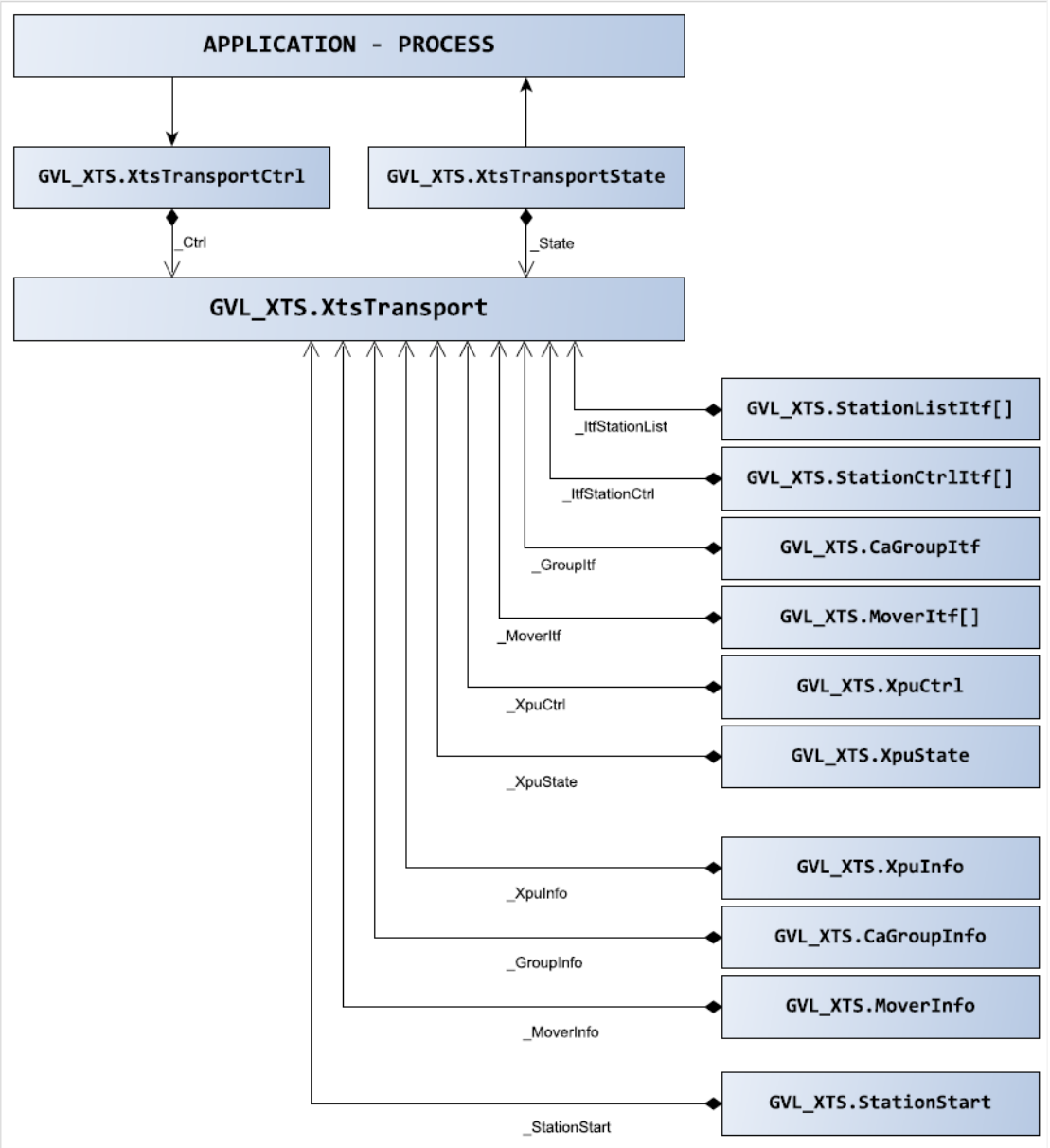
- TransportUnit
  - Fb_TransportUnit():
    - Top level control of XtsTransport
    - Cycle check for change of command:
      - **E_XTS_TRANSPORT_CTRL**.
        - CMD_INIT
        - CMD_IDLE
        - CMD_MOVER_ENABLE
        - CMD_MOVER_DISABLE
        - CMD_GROUP_CLEAR
        - CMD_GROUP_BUILD
        - CMD_GROUP_ENABLE
        - CMD_TRANSPORT_START



3. Design

BECKHOFF

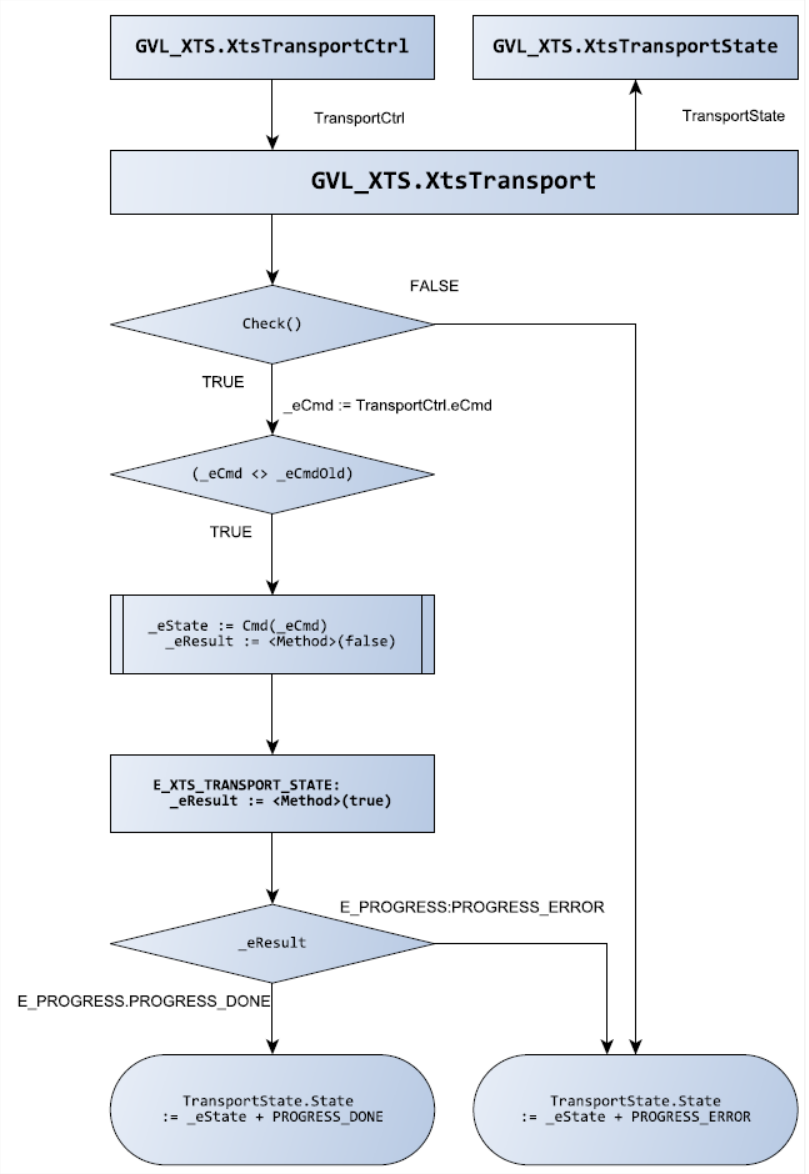
- TransportUnit
  - Fb_TransportUnit():
    - Members:



3. Design

BECKHOFF

- TransportUnit
  - Fb_TransportUnit():
    - Change of command triggers execution
    - Execution result is added to state
    - Extern control needs to react to BUSY, DONE or ERROR

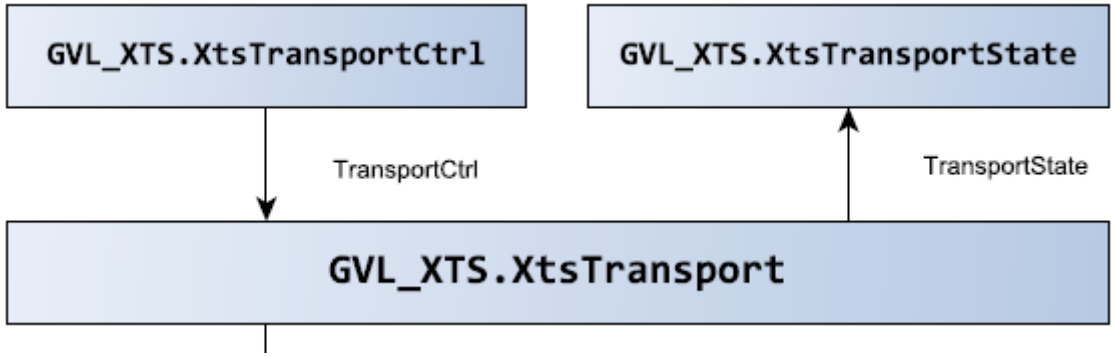


BECKHOFF

- TransportUnit
  - GVL_XTS.XtsTransportCtrl: **ST_TRANSPORT_UNIT_CTRL**
    - Struct for commanding FB_TransportUnit
    - eCmd : E_XTS_TRANSPORT_CTRL

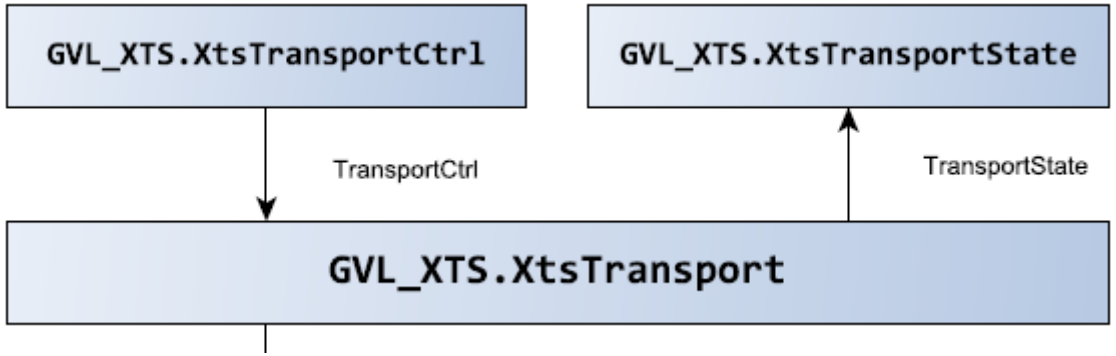
ST_XTS_TRANSPORT_CTRL

```
1  TYPE ST_XTS_TRANSPORT_CTRL :
2  STRUCT
3    Cmd        : E_XTS_TRANSPORT_CTRL;
4  END_STRUCT
5  END_TYPE
6
```



BECKHOFF

- TransportUnit
  - GVL_XTS.XtsTransportState: **ST_TRANSPORT_UNIT_STATE**
  - State: combines active command and result
  - Check: cyclic pointer checks
  - XpuState: state from fb_Xpu
  - XpuInfo: cyclic plausibility checks to TcCOM Objects
  - GroupInfo: cyclic information from FB_CaGroup



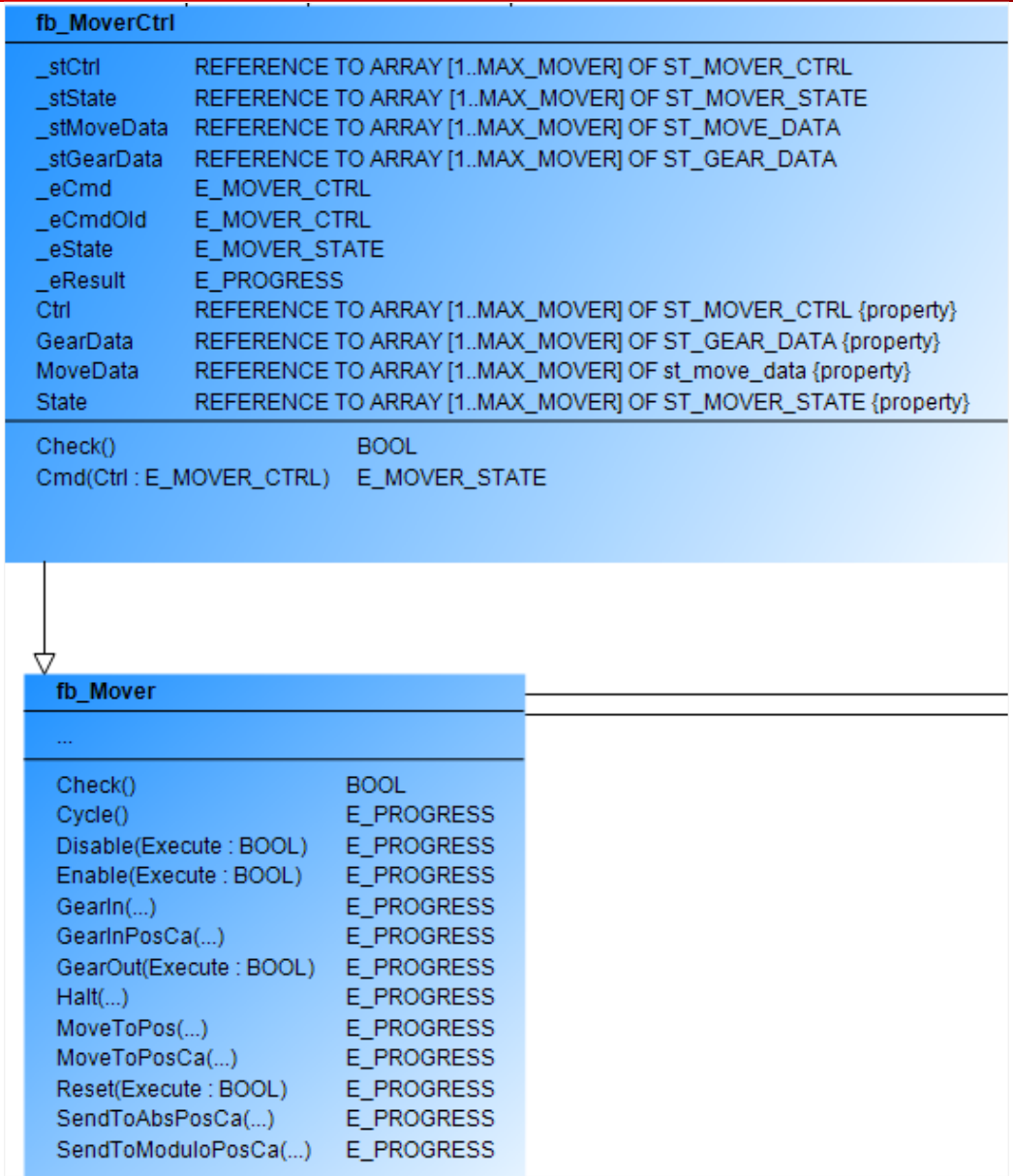
ST_XTS_TRANSPORT_STATE

```
1  {attribute 'pack_mode' := '2'}
2  TYPE ST_XTS_TRANSPORT_STATE :
3  STRUCT
4    State          : E_XTS_TRANSPORT_STATE;
5    Check          : E_XTS_TRANSPORT_CHECK;
6
7    XpuState       : ST_XPU_STATE;
8    XpuInfo        : ST_XPU_INFO;
9    GroupInfo      : ST_GROUP_INFO;
10 END_STRUCT
11 END_TYPE
12
```

3. Design

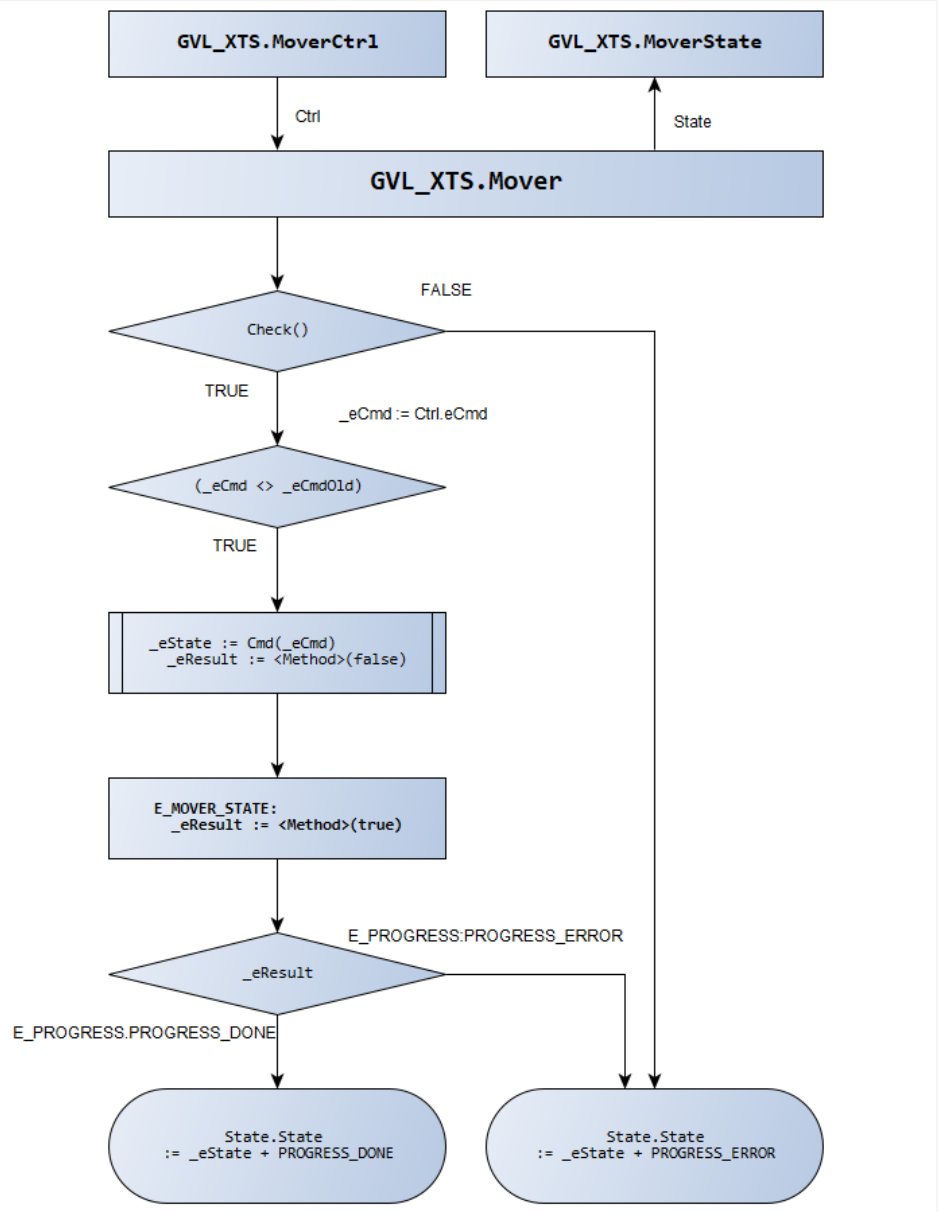
BECKHOFF

- GVL_XTS.Mover[] (**fb_MoverCtrl**)
  - Inherits **fb_Mover**
    - Access to MC function blocks in library
    - Implements Interface for use in other classes
  - Contains cyclic interface
    - OnChange check of command
    - Ctrl datafield for setting commands
    - State data field for checking responses
    - Parameter datafields for using motion functions



BECKHOFF

- fb_MoverCtrl:
  - Mover index is passed as value from caller
  - Global datafields are passed as references (**REF=**) into fb_MoverCtrl properties
    - OnChange Ctrl / State: handshakes
      - standard return value for method **(E_PROGRESS)**
    - OnExec log LastPosition of CA/MC function
    - OnExec log LastGap on CA function



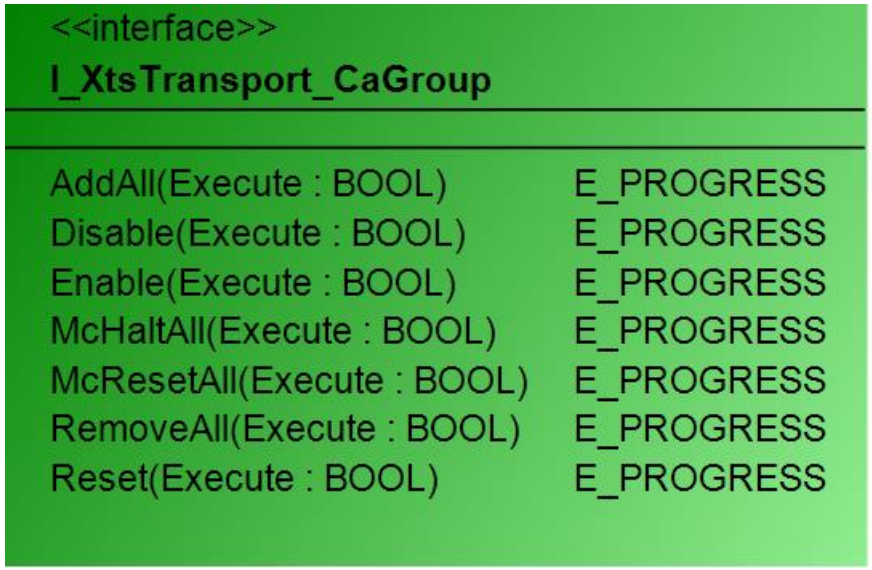
BECKHOFF

- fb_CaGroup:
  - Collision Avoidance class wrapper
  - Implements **I_Transport_CaGroup**
  - Cyclic information from **AXES_GROUP_REF**
  - Mover commands via interface **I_XtsTransport_Mover**

FB_CaGroup	
GROUP_HALT_JERK	LREAL
GROUP_HALT_DEC	LREAL
_eCheck	E_GROUP_CHECK
_bError	BOOL
_GroupRef	REFERENCE TO Tc3_McCoordinatedMotion.AXES_GROUP_REF
_GroupCommon	MCTOPLC_GROUP_COMMON_PART
_AxisRefMover	REFERENCE TO ARRAY [1..MAX_MOVER] OF Tc2_MC2.AXIS_REF
_MoverItf	REFERENCE TO ARRAY [1..MAX_MOVER] OF I_XtsTransport_Mover
_stMoveData	ST_MOVE_DATA
_fbAddAxisGroup	ARRAY [1..MAX_MOVER] OF Tc3_McCoordinatedMotion.MC_AddAxisToGroup
_fbRemoveAxisGroup	ARRAY [1..MAX_MOVER] OF Tc3_McCoordinatedMotion.MC_RemoveAxisFromGroup
_fbGroupDisable	Tc3_McCoordinatedMotion.MC_GroupDisable
_fbGroupEnable	Tc3_McCoordinatedMotion.MC_GroupEnable
_fbGroupErrorRead	Tc3_McCoordinatedMotion.MC_GroupReadError
_fbGroupStatusRead	Tc3_McCoordinatedMotion.MC_GroupReadStatus
_fbGroupReset	Tc3_McCoordinatedMotion.MC_GroupReset
_stGroupInfo	ST_GROUP_INFO
_rtrigGroupStatusRead	Tc2_Standard.R_TRIG
_rtrigGroupErrorRead	Tc2_Standard.R_TRIG
_stMsg	ST_Message
_eMessageLevel	E_MessageType
AxisRef	REFERENCE TO ARRAY [1..MAX_MOVER] OF Tc2_MC2.AXIS_REF {property}
GroupInfo	REFERENCE TO ST_GROUP_INFO {property}
GroupRef	REFERENCE TO Tc3_McCoordinatedMotion.AXES_GROUP_REF {property}
MessageLevel	e_messagetype {property}
MoverItf	REFERENCE TO ARRAY [1..MAX_MOVER] OF I_XtsTransport_Mover {property}
...	

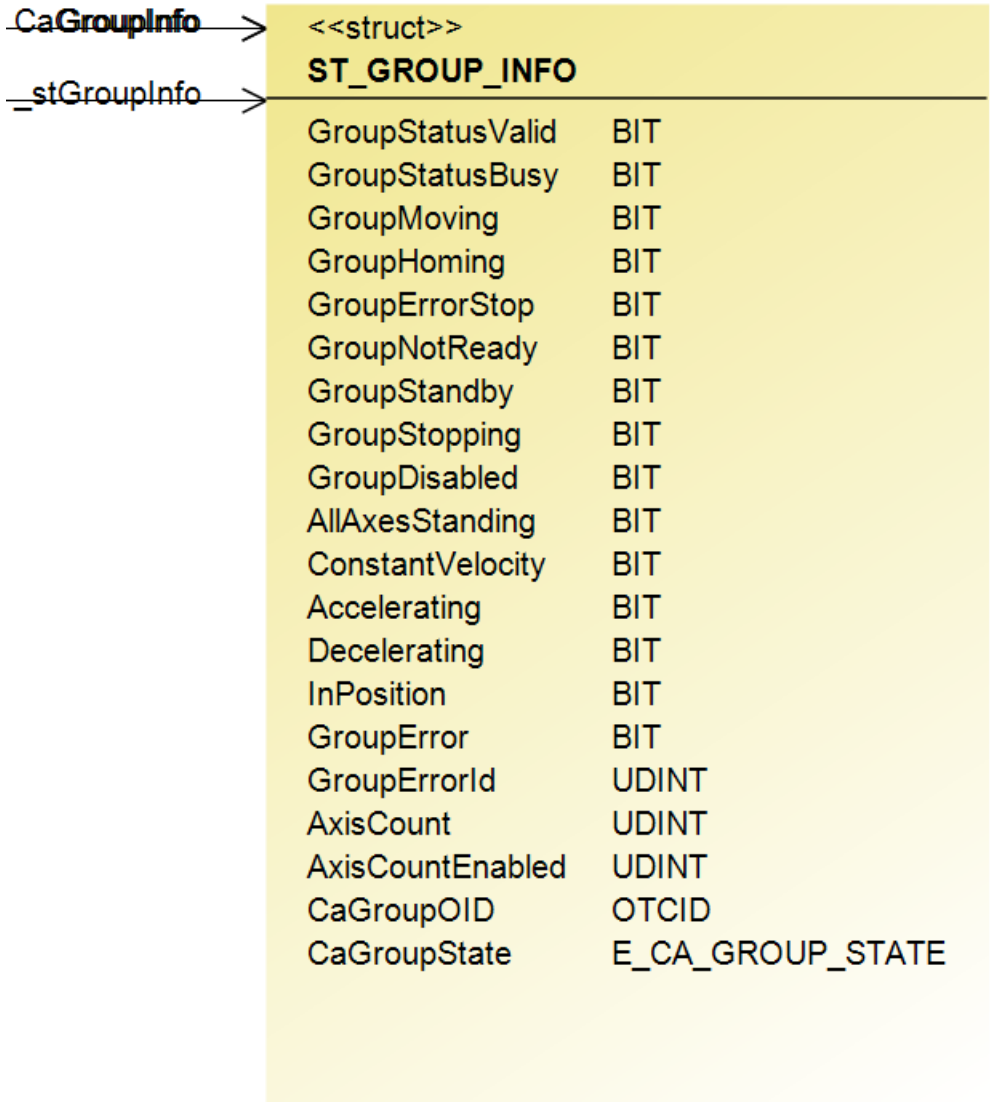
BECKHOFF

- fb_CaGroup:
  - Implements **I_Transport_CaGroup**
    - Used in fb_TransportUnit



BECKHOFF

- fb_CaGroup:
  - Cyclic information to **ST_GROUP_INFO**



BECKHOFF

- GVL_XTS.Xpu (**fb_XpuCtrl**)
  - Inherits **fb_Xpu**:
    - Class for interacting with XTS ProcessingUnit
    - XpuInit()
      - Connects to OTCIDs of XTS TcCOM Objects
    - Cycle
      - Plausibitlity checks, get module info data
    - ModuleInfoData, used in Cycle

fb_XpuCtrl

_Ctrl	REFERENCE TO ST_XPU_CTRL
_State	REFERENCE TO ST_XPU_STATE
_eCmd	E_XPU_CTRL
_eCmdOld	E_XPU_CTRL
_eResult	E_PROGRESS
_eState	E_XPU_STATE
Ctrl	REFERENCE TO ST_XPU_CTRL {property}
State	REFERENCE TO ST_XPU_STATE {property}

Check()	BOOL
Cmd(Ctrl : E_XPU_CTRL)	E_XPU_STATE
DetectMoverId(Enable : BOOL)	E_XPU_CHECK

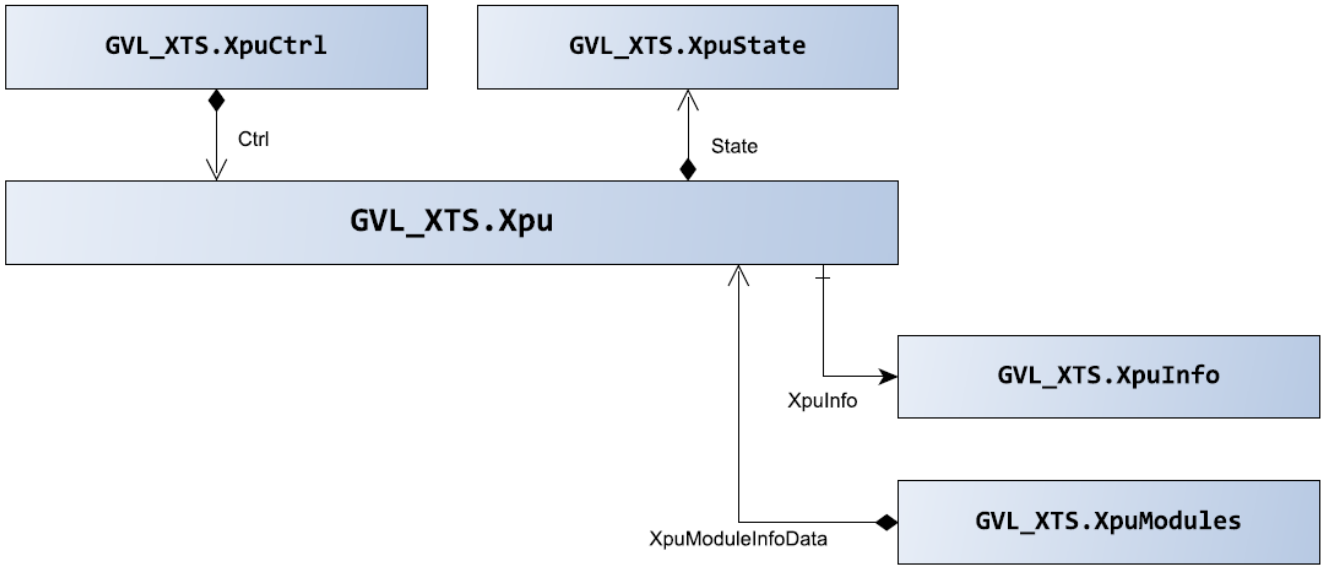
fb_Xpu

...

Cycle(...)	E_PROGRESS
GetEnvironment()	I_TcIoXtsEnvironment
IdDetectionModeToString()	STRING(20)
ModuleInfoData(Enable : BOOL)	E_PROGRESS
MoverPositionAssignementToString()	STRING(20)
OpModeToString()	STRING(20)
XpuInit(...)	E_XPU_INIT

BECKHOFF

- GVL_XTS.Xpu (**fb_XpuCtrl**)
  - Wraps cyclic execution of fb_Xpu
  - Cyclic check for command change **(ST_XPU_CTRL.Cmd)**



E_XPU_CTRL

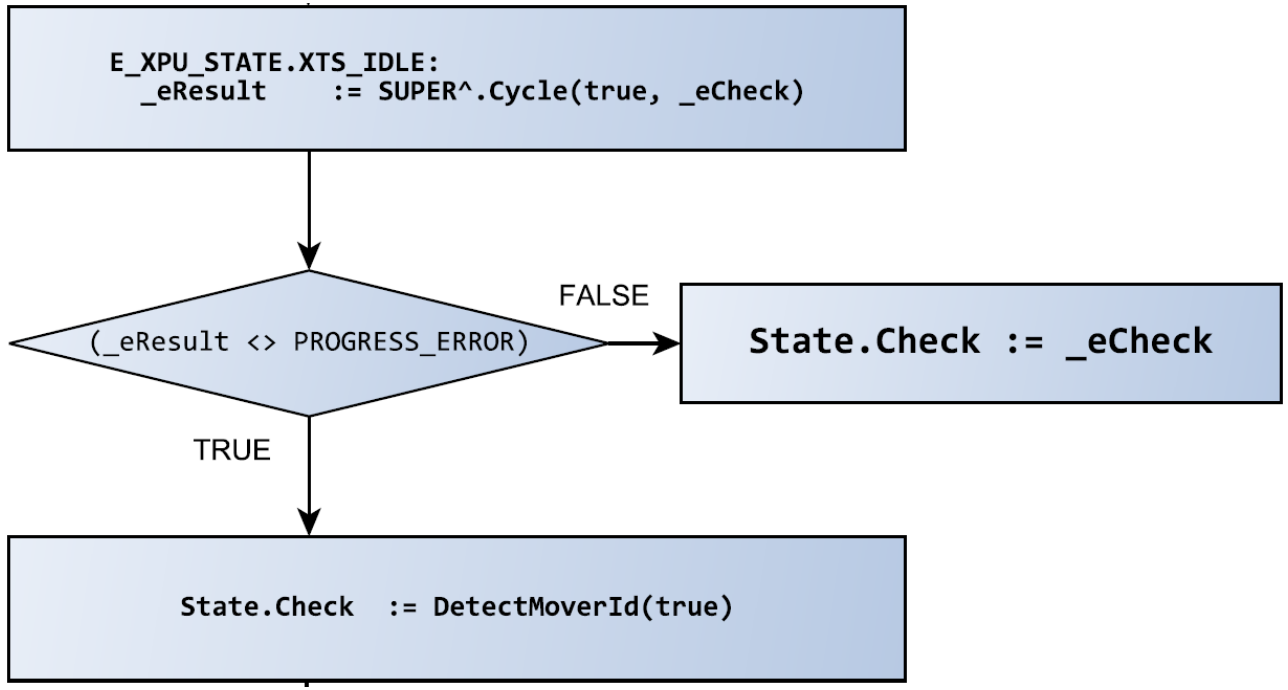
```
{attribute 'strict'}
TYPE E_XPU_CTRL :
(
  XTS_CMD_NULL,
  XTS_CMD_INIT                := 10,
  //XTS_CMD_READ_STN_CONFIG     := 20,
  XTS_IDLE                    := 30
)UINT;
END_TYPE
```

BECKHOFF

- fb_XpuCtrl cyclic data:
  - **SUPER^.Cycle():**
    - Cyclic plausibility checks
    - Cyclic update motor modules data
  - Cyclic data **ST_XPU_INFO:**

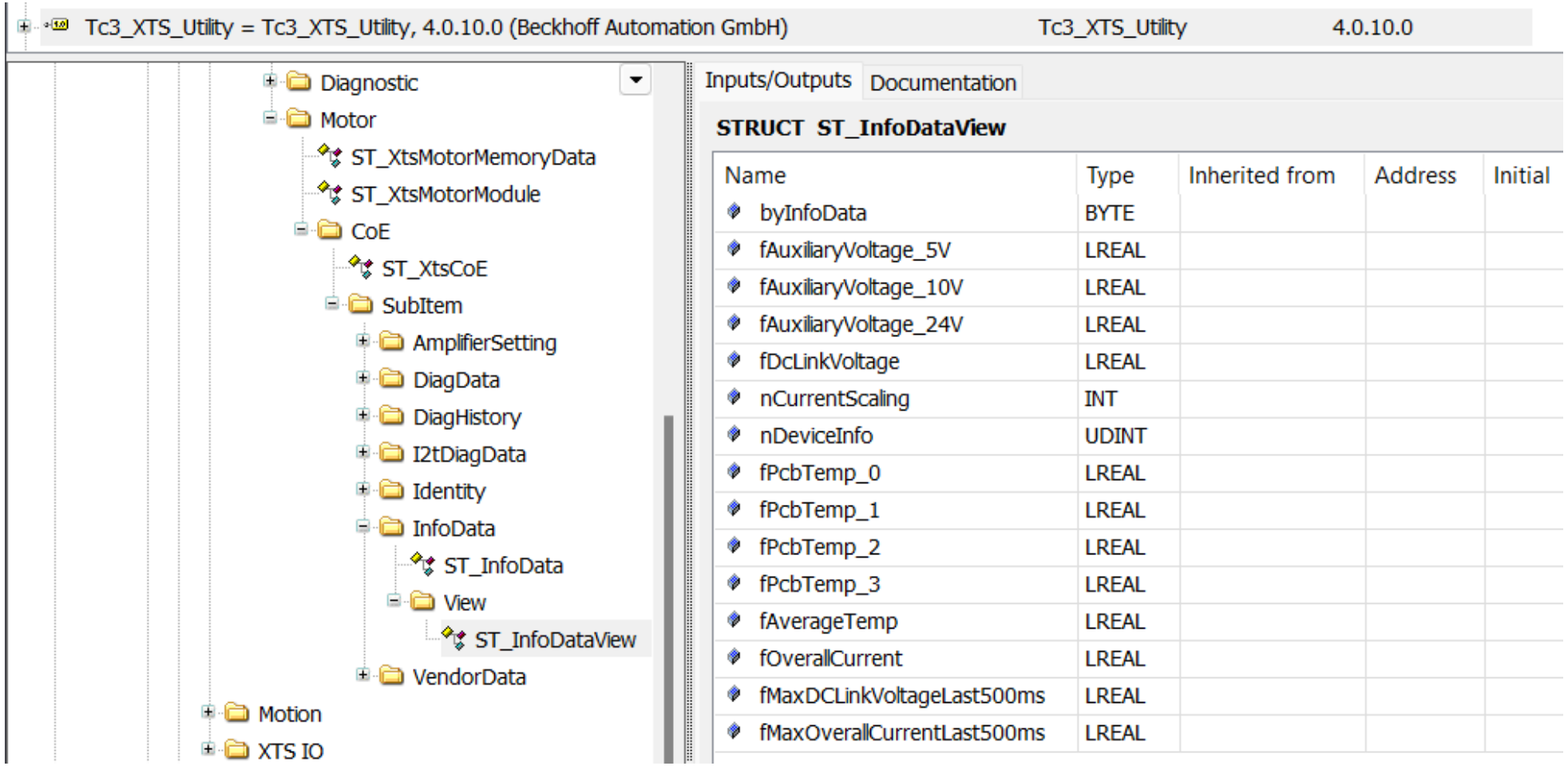
ST_XPU_INFO

```
{attribute 'pack_mode' := '2'}
TYPE ST_XPU_INFO :
STRUCT
  AllPositionsValid    : BIT;
  IdDetectionError     : BIT;
  IdDetectionValid     : BIT;
  IdDetectionActive    : BIT;

  OperationMode        : UINT;

  IdDetectionMode           : UINT;
  MoverPositionAssignement  : UINT;

  nDetectedAxisCount   : UINT;
  nExpectedAxisCount   : UINT;


END_STRUCT
END_TYPE
```



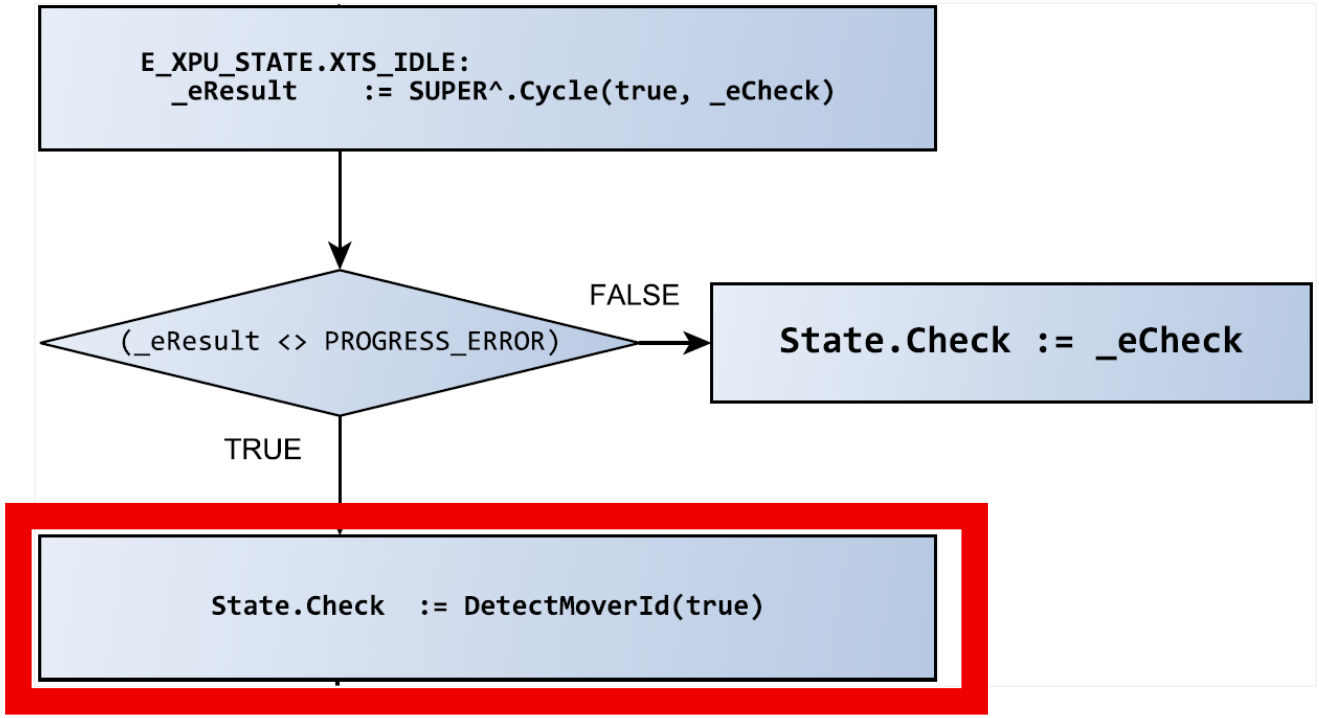
BECKHOFF

- fb_XpuCtrl:
  - Cyclic motor module data:



BECKHOFF

- fb_XpuCtrl cyclic data:
  - **DetectMoverId:**
    - Cyclic checks for valid Mover ID Detection
    - Check pdf flowchart in [doc] folder of project



XTS_TRANSPORT_LAYER project