XTS TRANSPORT LAYER – a station based approach

BECKHOFF

XTS TRANSPORT LAYER (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

XTS TRANSPORT LAYER (HAUD)

BECKHOFF

1. Introduction

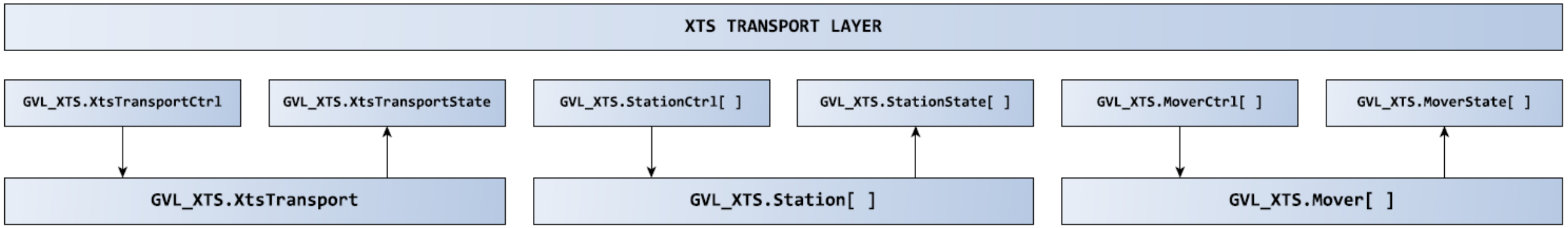
- functional basics of CA Group
- use of XTS_Utility lib
- introduction to station based approach
- configurable station placement
- station acts as sender
- configurable station design with basic transport logic
- configurable station design with variable transport logic
- individual targeting of mover to Station
- grouping of stations for parallel or serial work flow
- function blocks with ctrl/state structs:
  - cyclic check on change of command enumeration
  - state struct enumeration with offsets for progress
- 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 have an interface for guiding a mover through a process station
- A transport layer shall have an interface to manipulate a mover within a station or for a certain task
- A transport layer shall have an interface for setting-up or clearing the CollisionAvoidance Group

XTS TRANSPORT LAYER (HAUD)

BECKHOFF

1. Introduction

- designed for use with extern cyclic or non cyclic flow control
- station based approach with individual targeting 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)



1. Introduction

Planning requirements for use of fb_Station:

- Put the Modulo turn anywhere, **BUT NOT** within WaitPos, StopPos, ReleaseDistance of a station. The code does not support crossing the modulo turn within a station.

- The Use of LinkedList methods (AddTail, GetHead) requires thought about when the mover is entered into the target station.
- a. parallel stations for a process:
    - P1 uses XTS_STN[1] to XTS_STN[4] → rReleaseDistance of STN[4] shall be shortest, all other stations follow accordingly.
- b. 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: STATION_MOVER_SEND.
- c. 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
        - Do not send any new mover to the station in question
            - disable station
            - preceeding stations continue workflow with changed ST_STATION_CTRL.nTargetStation

know thyself

1. Introduction

Planning requirements for use of fb_Station:

- know thyself
    - 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 you have to make sure that there is room for it
      --> distance between PosWait and PosStop

2. Requirements

BECKHOFF

- XtsTransport
  - Access to CA group function blocks (interface pointer)
  - Access to Stations (interface pointer)
  - Access to Movers (interface pointer)
  - 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

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
  - Interface pointer for access from:
    - TransportUnit
    - Station

2. Requirements

BECKHOFF

- Station
  - Handshake mover transport with extern control
  - Close observation of movements with feedback to extern control
  - List for movers in queue
  - Cyclic interface for access from extern control
    - Ctrl (write): command and parameter
    - State (read): response to command and information about mover and queue
  - Uses Mover interface pointer

3. Design

BECKHOFF

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

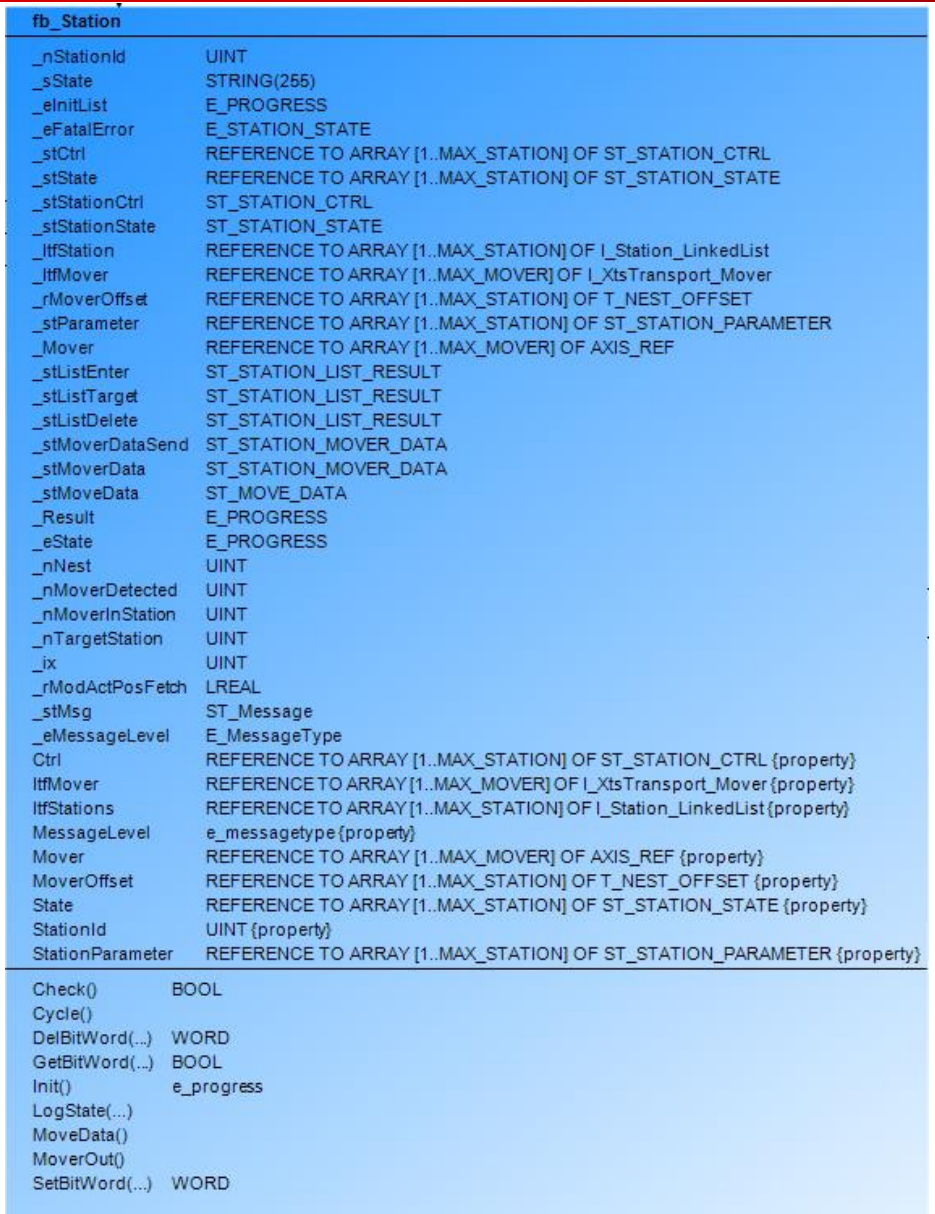
<<global>>
GVL_XTS

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

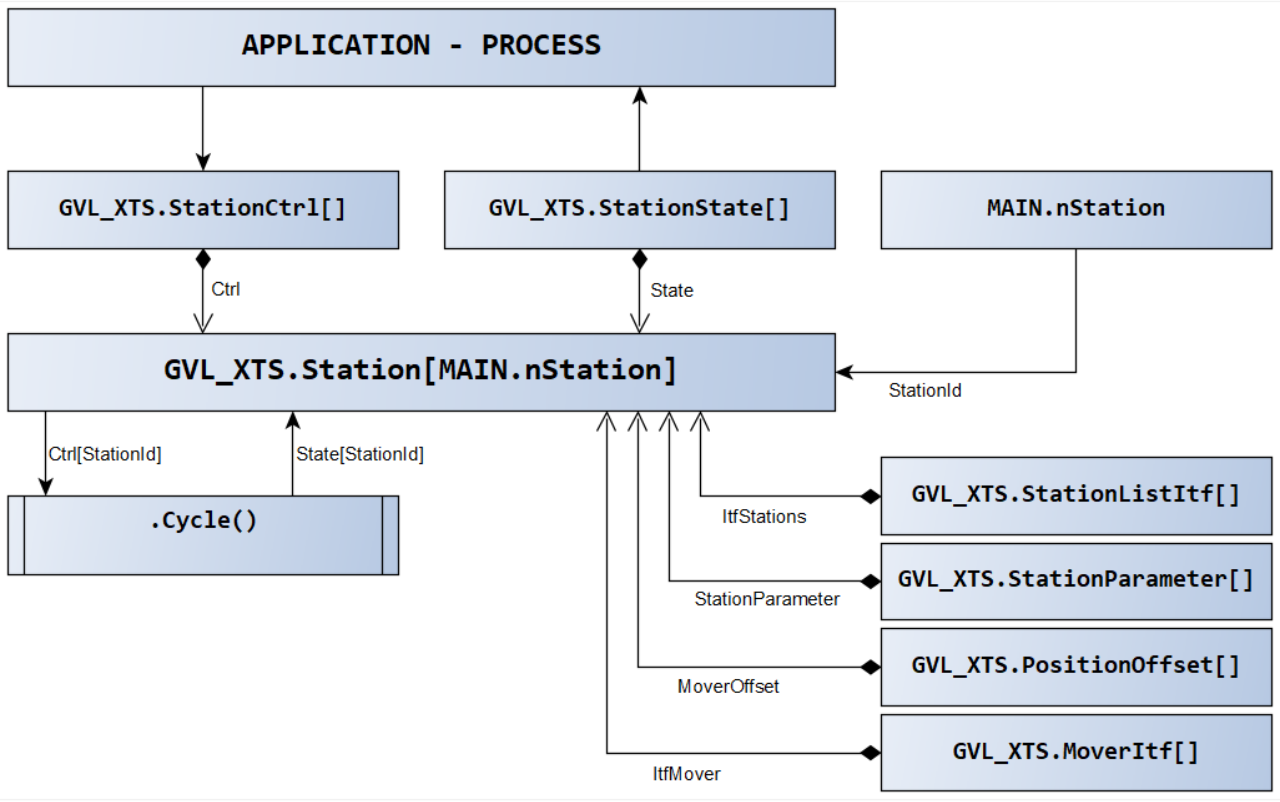
- GVL_XTS.Station
  - fb_Station[].Cycle
    - State machine for handshaking with extern control
      - Init (clears everything in station)
      - Enable
      - Mover Enter
      - Stop Position(s)
      - Mover Out
      - Empty
    - Control writes ticket for mover
      - MoverId
      - TargetStation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



3. Design

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



3. Design

BECKHOFF

- GVL_XTS.Station
  - Ctrl[nStation] : ST_STATION_CTRL
    - eCmd :
      - enumeration for handshakes with State[nStation].eState
    - nMask :
      - 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 :
      - target to send mover to GVL_XTS.Station[nTargetStation].WaitPos
    - rOffset :
      - Optional offset for mover, used in target station in addition to static offset

3. Design

BECKHOFF

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

3. Design

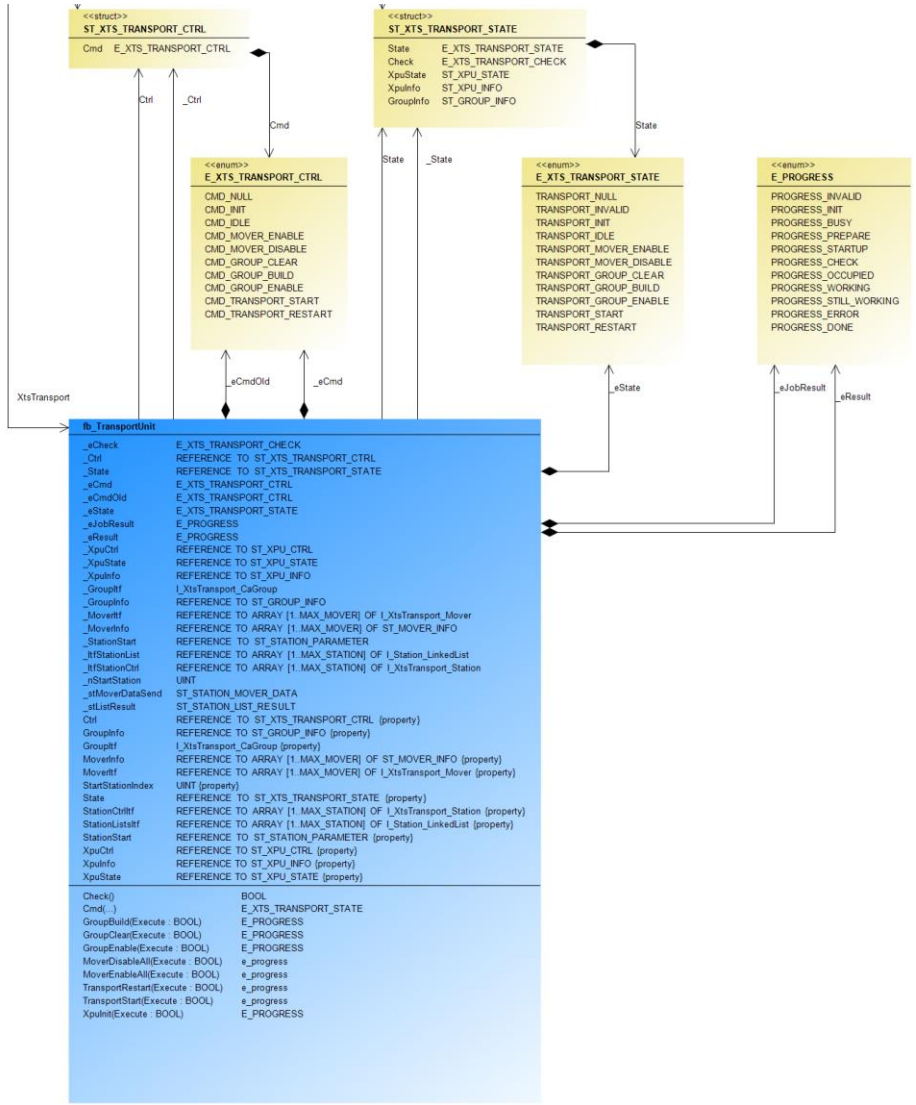
BECKHOFF

- 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 has to stop at in XtsStation
  - rPosStop[] :
    - Relative to rPosWait

3. Design

BECKHOFF

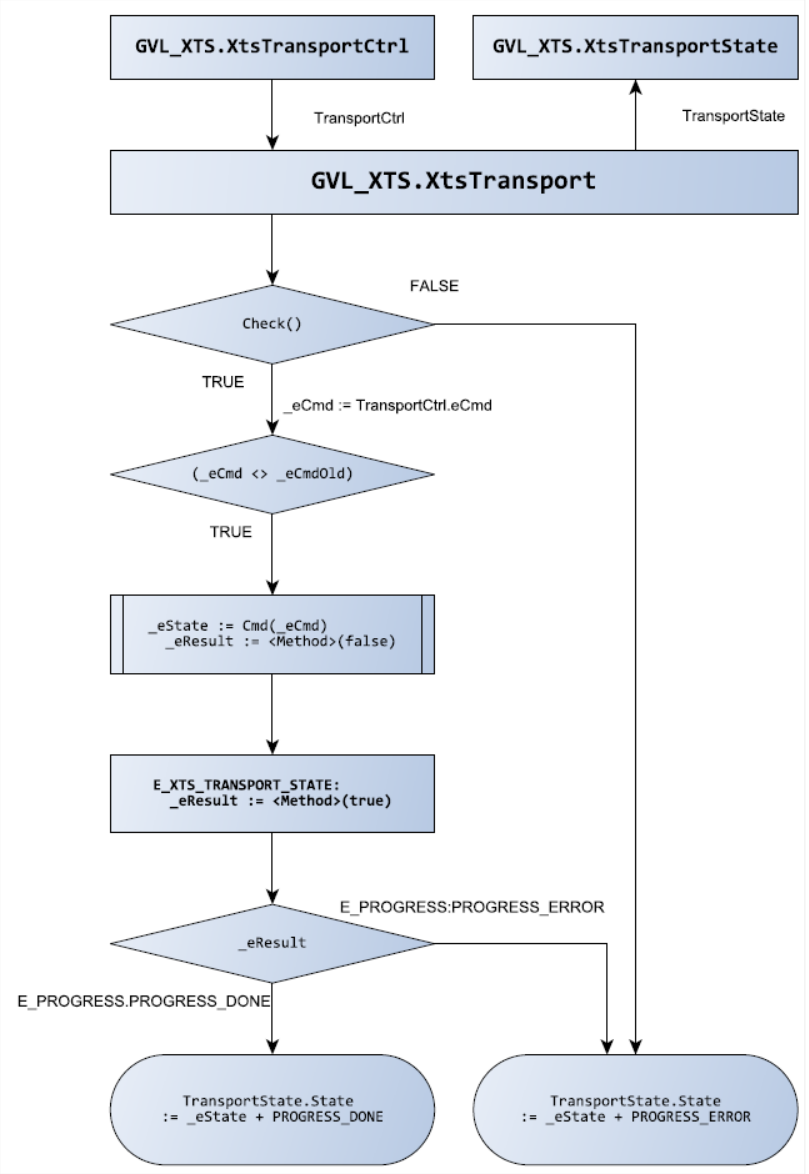
- TransportUnit
  - Fb_TransportUnit():
    - Top level control of XtsTransport
    - Cycle check for change of command:
      - E_XTS_TRANSPORT_CTRL.
        - CMD_INIT
        - CMD_IDLE
        - CMD_MOVER_ENABLE
        - CMD_MOVER_DISABLE
        - CMD_GROUP_CLEAR
        - CMD_GROUP_BUILD
    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



3. Design

BECKHOFF

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

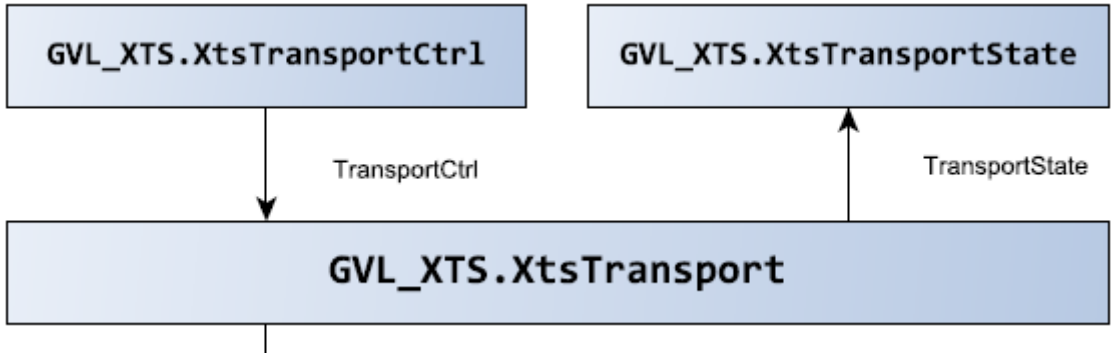


BECKHOFF

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

ST_XTS_TRANSPORT_CTRL

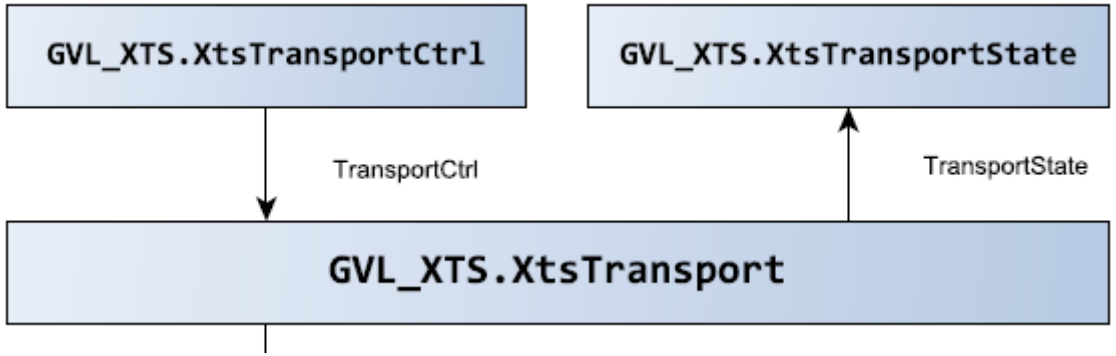
```
1  TYPE ST_XTS_TRANSPORT_CTRL :
2  STRUCT
3    Cmd        : E_XTS_TRANSPORT_CTRL;
4  END_STRUCT
5  END_TYPE
6
```



3. Design

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



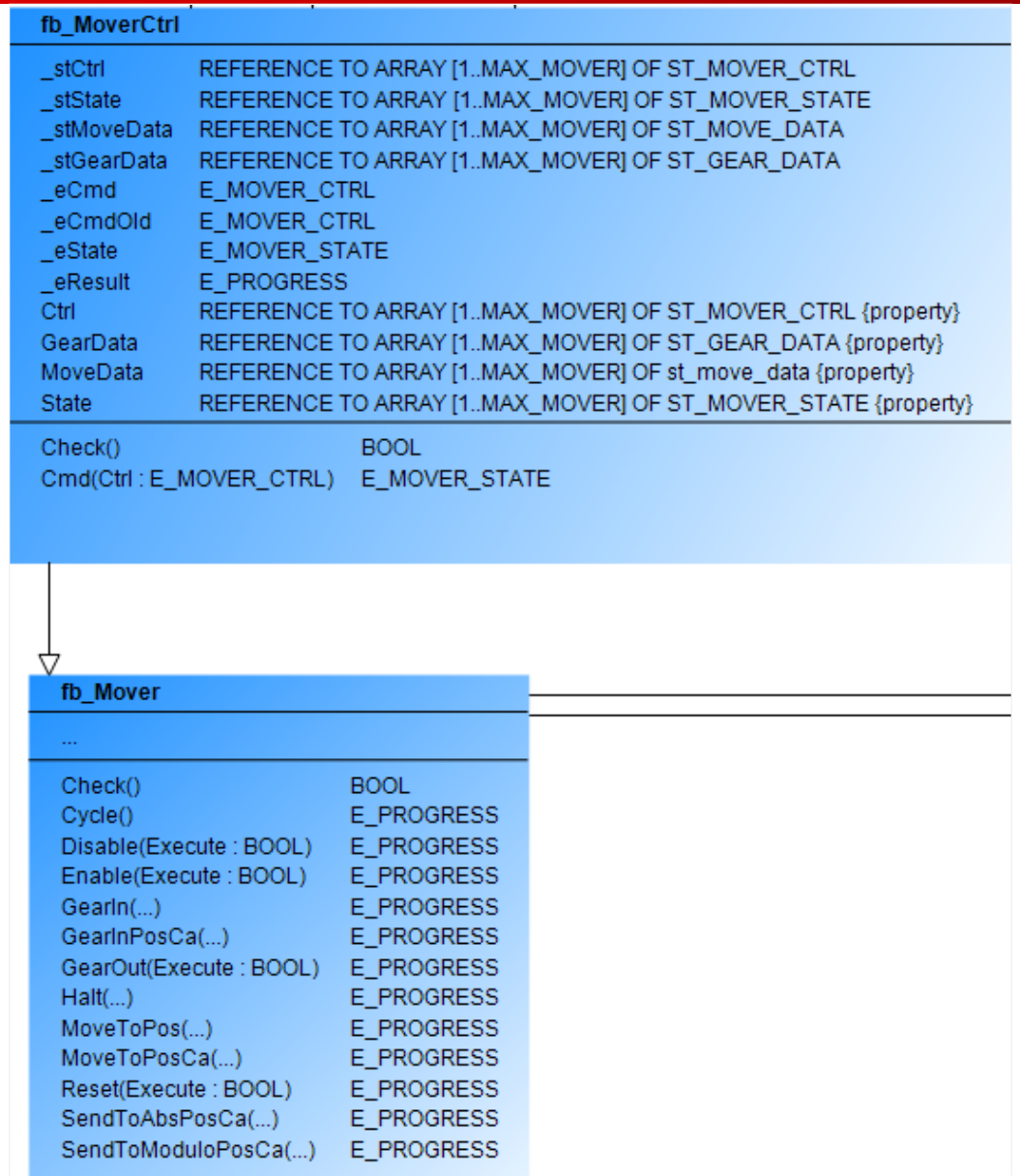
ST_XTS_TRANSPORT_STATE

```
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

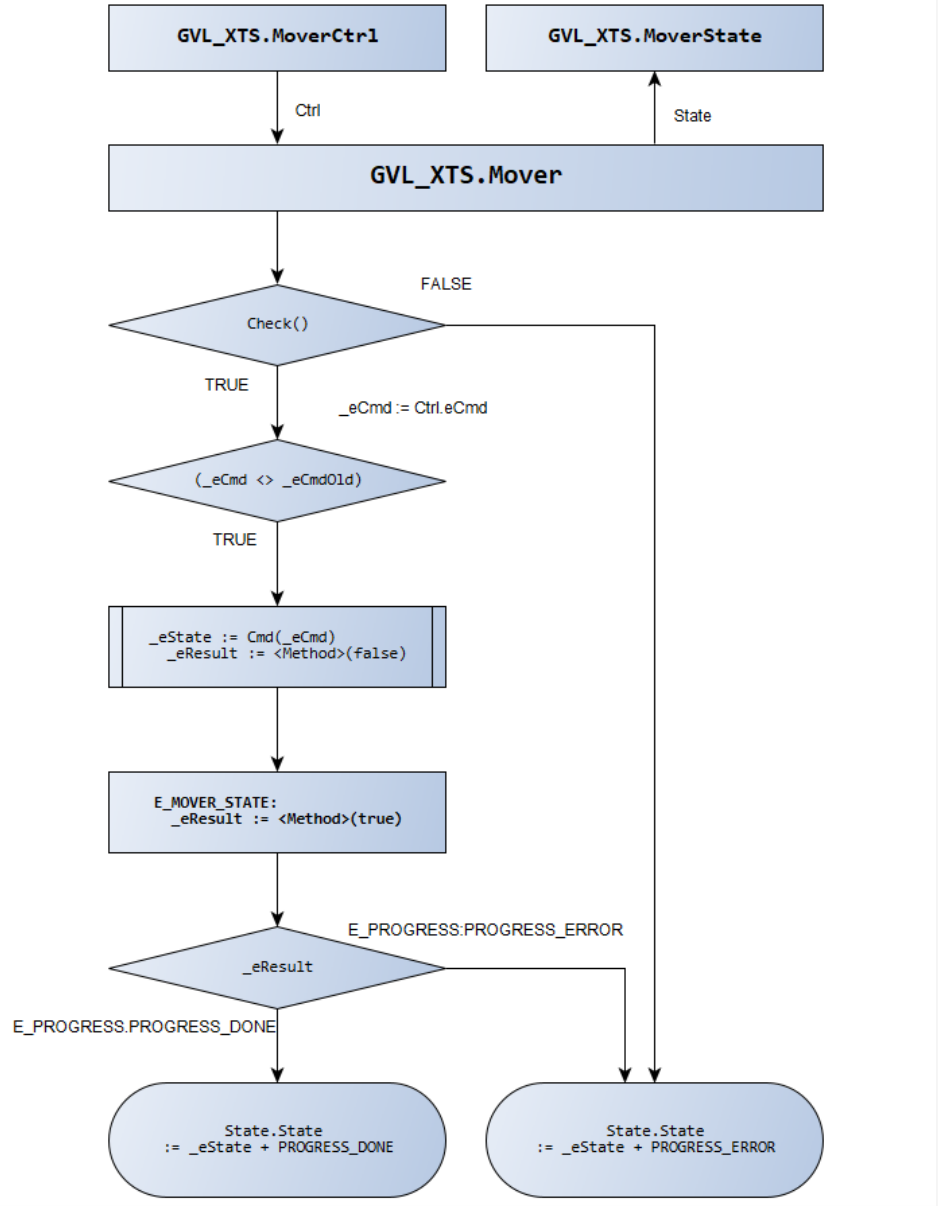
- fb_MoverCtrl:
  - Inherits fb_Mover
    - Access to MC function blocks in library
    - Implements Interface for use in other classes
  - Contains cyclic interface
    - Ctrl datafield for setting commands
    - State data field for checking responses
    - Parameter datafields for using motion functions



3. Design

BECKHOFF

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



XTS_TRANSPORT_LAYER project