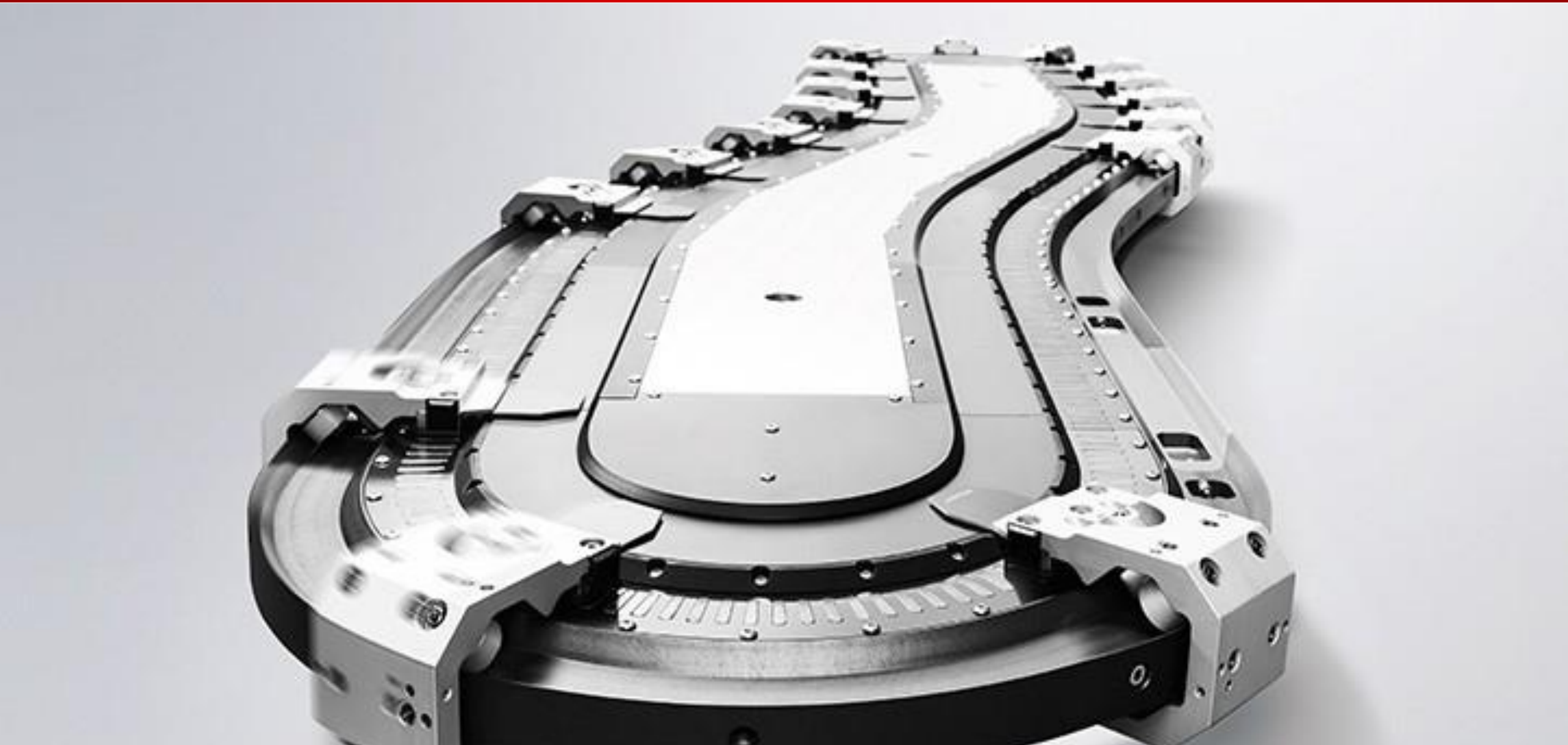
XTS TRANSPORT LAYER – Mover Class

BECKHOFF



XTS TRANSPORT LAYER – Mover Class

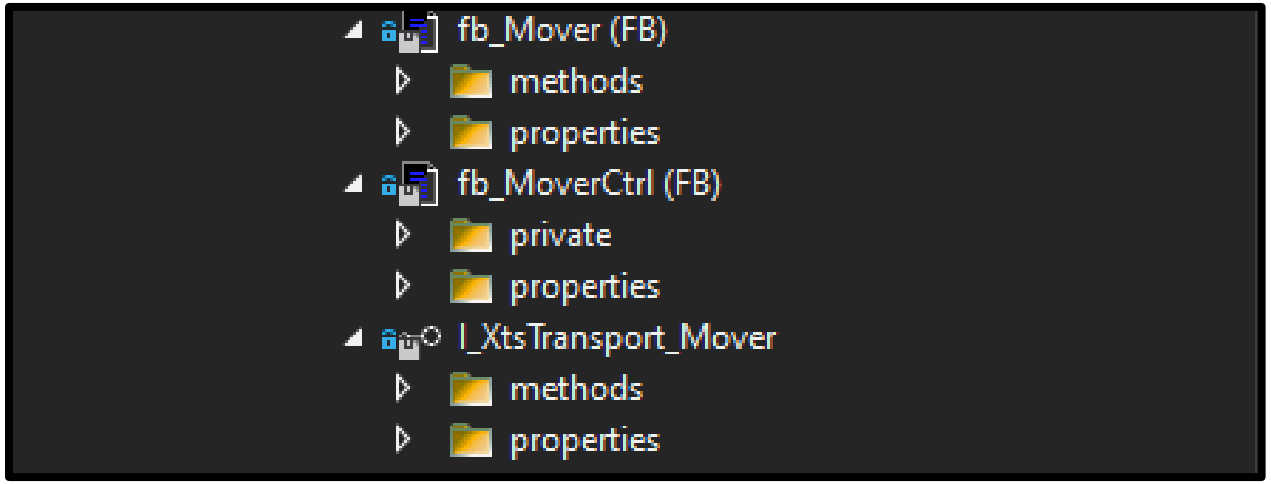
BECKHOFF

1. Design

Design

BECKHOFF

- **GVL_XTS.Mover (fb_MoverCtrl)**
  - fb_Mover
    - Encapsulates Motion Control and Collision Avoidance function blocks
    - Message handling
    - Base class of fb_MoverCtrl
  - fb_MoverCtrl
    - Cyclic execution wrapper
    - OnChange check of commands
    - State feedback
  - I_XtsTransport_Mover
    - Interface for use in fb_CaGroup, fb_Station, fb_TransportUnit

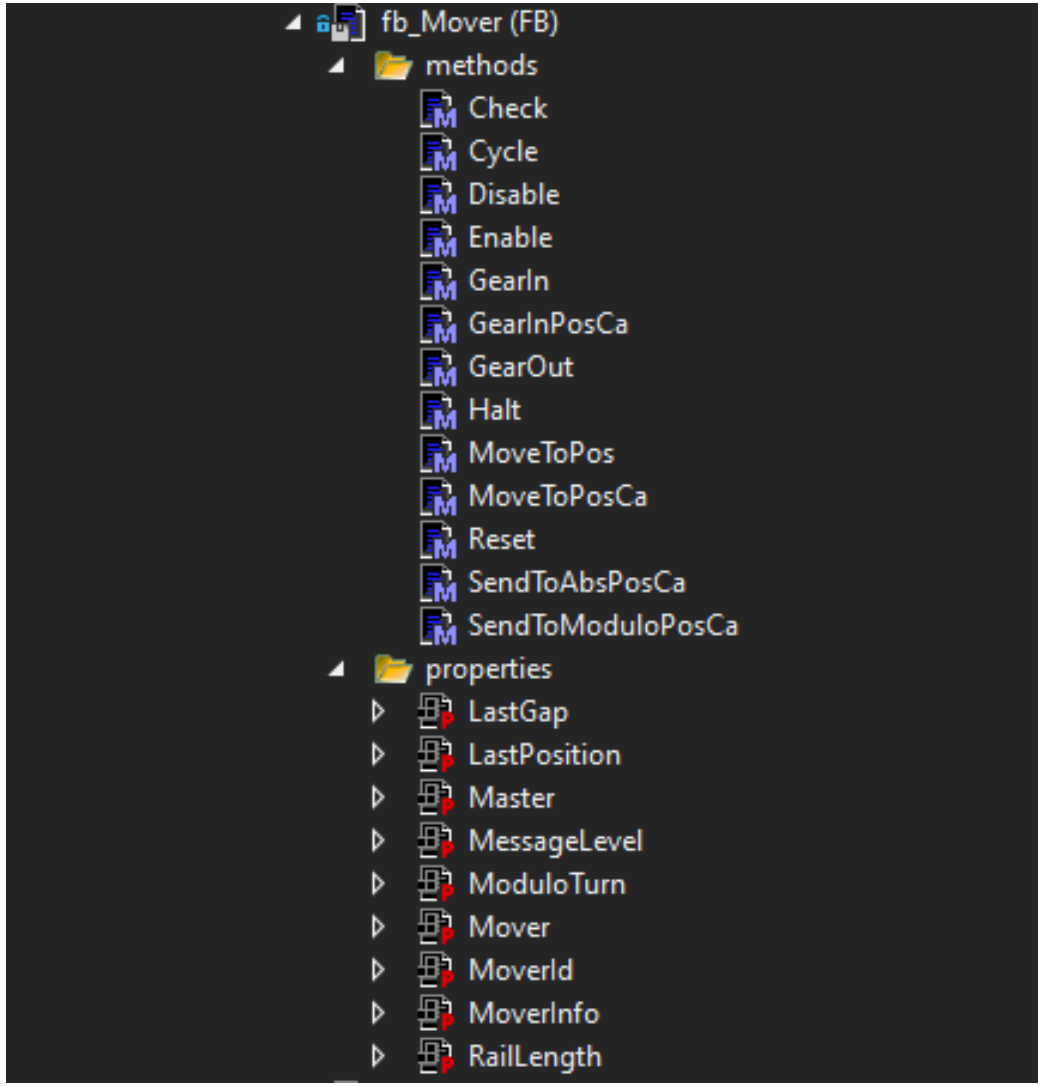


Design

BECKHOFF

- **fb_Mover:**
  - **Cycle()** method must be called by extending class
  - **Check()** must be used in extending class (pointer checks)
  - Method flow charts, see doc folder
    - Execute behaviour:

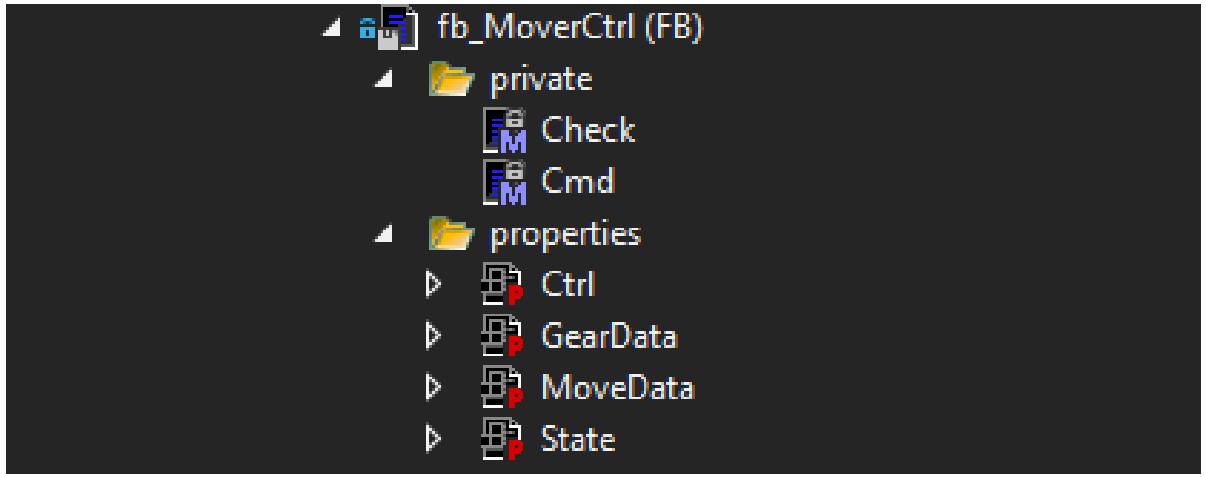
      → initialize call with ‘Execute:=FALSE’
    - Methods feedback: E_PROGRESS



Design

BECKHOFF

- **fb_MoverCtrl():**
  - Requires cyclic call of instance
    - See calls in MAIN() as example
  - OnChange check of Ctrl.Cmd
  - References to Ctrl/State structs
  - References to Mover motion parameter structs
  - State.State feedback:
    - Command enum equivalent + E_PROGRESS



BECKHOFF

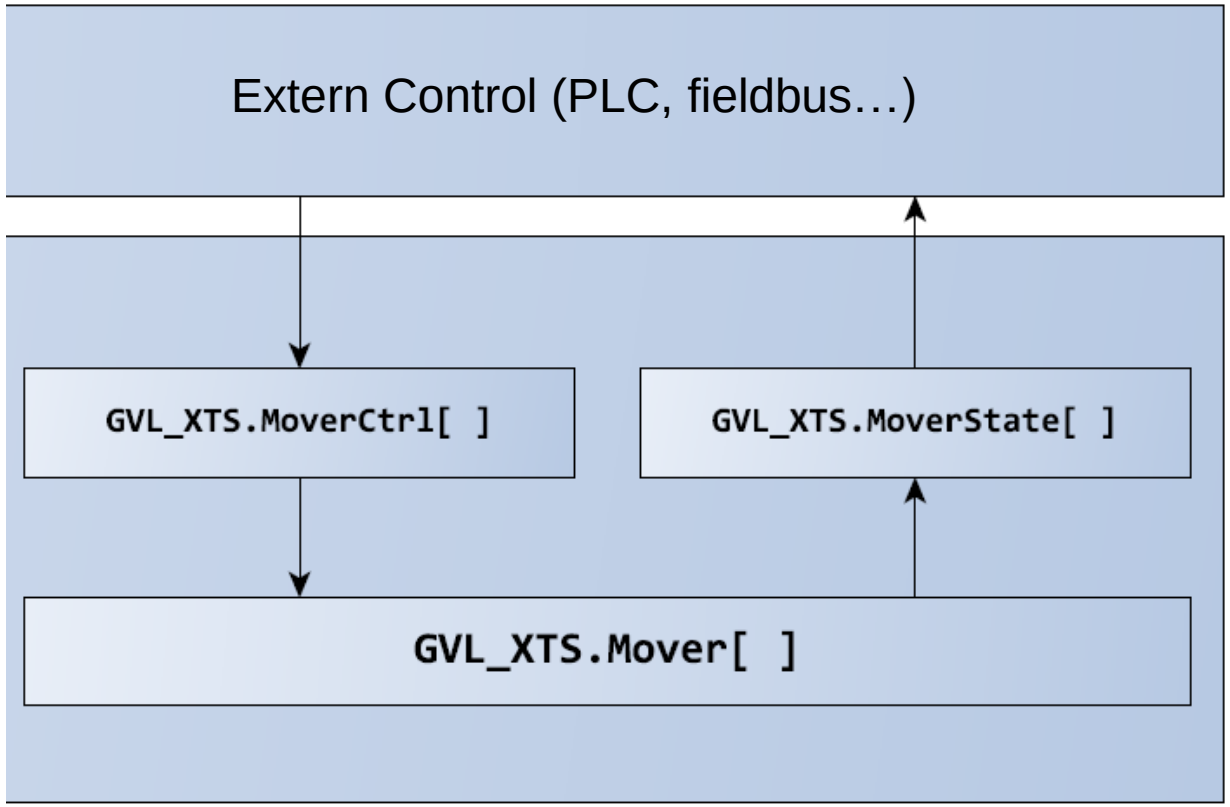
- **fb_MoverCtrl():**
  - Cyclic execution of movement commands
  - OnChange check of Ctrl.Cmd
  - References to Ctrl/State structs
  - References to Mover motion parameter structs

```
21  FUNCTION_BLOCK fb_MoverCtrl EXTENDS fb_Mover
22  VAR
23    // Ctrl and State for executing commands
24    _stCtrl           : REFERENCE TO ARRAY[1..MAX_MOVER] OF ST_MOVER_CTRL;
25    _stState          : REFERENCE TO ARRAY[1..MAX_MOVER] OF ST_MOVER_STATE;
26
27    // data structures to use with commands
28    _stMoveData       : REFERENCE TO ARRAY[1..MAX_MOVER] OF ST_MOVE_DATA;
29    _stGearData       : REFERENCE TO ARRAY[1..MAX_MOVER] OF ST_GEAR_DATA;
30
31    _eCmd,
32    _eCmdOld          : E_MOVER_CTRL;
33
34    _eState           : E_MOVER_STATE;
35
36    _eResult          : E_PROGRESS;
37  END_VAR
```

```
1   IF NOT Check() THEN RETURN; END_IF
2
3   _stMsg.eDevice                := e_Device.Mover + _nMoverId;
4
5   // copy to local for debug
6   _eCmd                         := _stCtrl[_nMoverId].Cmd;
7
8   // cyclic check for command change
9   // get state for cmd
10  IF (_eCmd <> _eCmdOld)
11  THEN
12    // get matching sdtate for Ctrl.Cmd
13    _eState                     := Cmd(_eCmd);
14    _eCmdOld                    := _eCmd;
15  END_IF
```

BECKHOFF

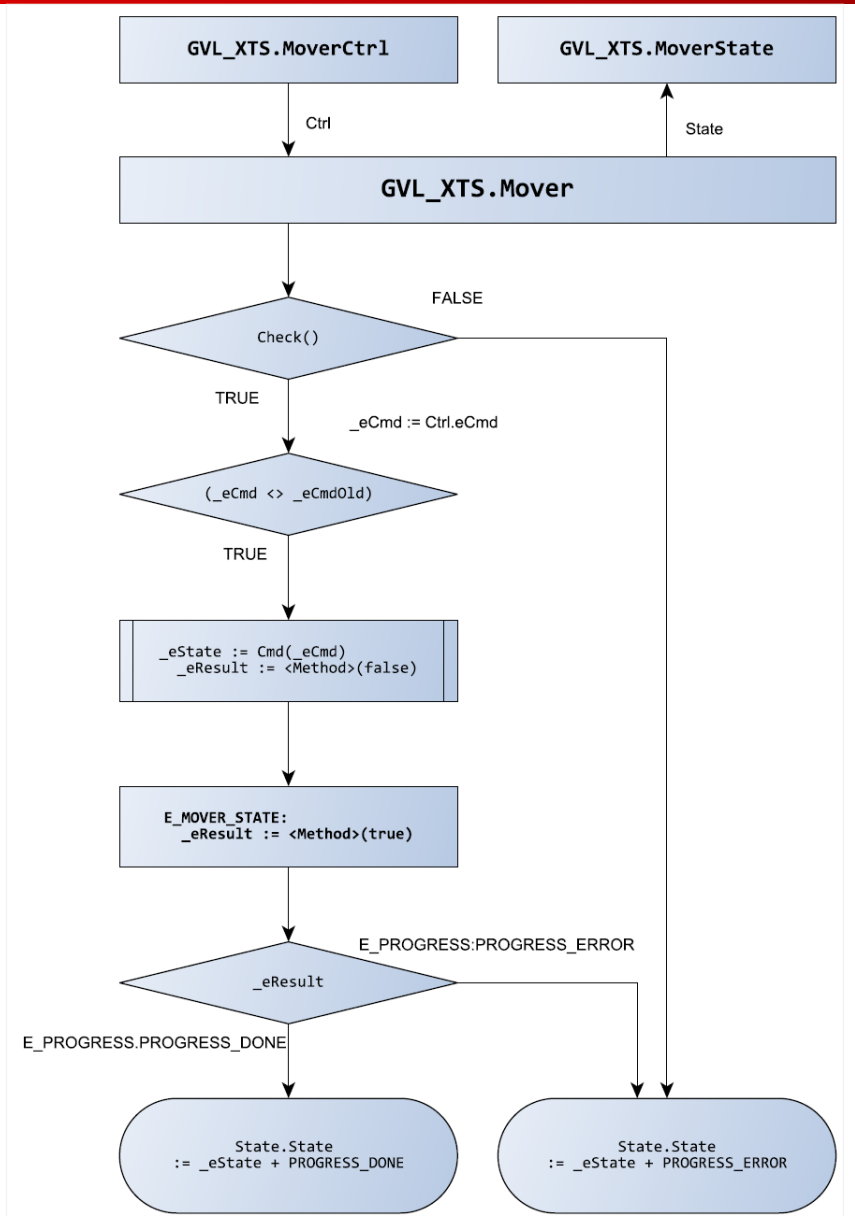
- **fb_MoverCtrl():**
  - Control structure
    - ST_MOVER_CTRL
  - State structure
    - ST_MOVER_STATE
  - Ctrl/State structs are used to address mover
  - Parameter structures are to be used according to the job you want to give to the mover
    - ST_MOVE_DATA
    - ST_GEAR_DATA
  - Mover information:
    - ST_MOVER_INFO



Design

BECKHOFF

- **fb_MoverCtrl():**
  - **ST_MOVER_CTRL / ST_MOVER_STATE**
  - Command
  - State + Progress
    - See doc folder for flow chart



BECKHOFF

- **fb_MoverCtrl():**
  - **ST_MOVE_DATA**
  - Parameter struct for motion commands

XTS_TRANSPORT | ST_MOVE_DATA | TC_XTS_BASE

```
1  {attribute 'pack_mode' := '2'}
2  TYPE ST_MOVE_DATA :
3  STRUCT
4    // HAUD 2024 03 14
5    // some init data added for example
6    rPos             : LREAL :=   2275.0;
7    rVelo            : LREAL :=    500.0;
8    rAcc             : LREAL :=   5000.0;
9    rJerk            : LREAL := 500000.0;
10   rGap             : LREAL :=    100.0;
11   rOverride        : LREAL :=    100.0;
12   rDelta           : LREAL :=      0.05;
13   rDistance        : LREAL :=     25.0;
14 END_STRUCT
15 END_TYPE
16
```

BECKHOFF

- **fb_MoverCtrl():**
  - **ST_GEAR_DATA**
  - Parameter struct for motion commands
    - E_MOVER_CTRL.MOVER_GEAR_IN
    - E_MOVER_CTRL.MOVER_GEAR_OUT
    - E_MOVER_CTRL.MOVER_GEAR_IN_POS_CA

XTS_TRANSPORT | ST_GEAR_DATA | E_MOVER_CTRL

```
1  {attribute 'pack_mode' := '2'}
2  TYPE ST_GEAR_DATA :
3  STRUCT
4    rDeltaToMasterPos        : LREAL;
5    rModuloSyncPosSlave      : LREAL;
6    rMasterStartDistance     : LREAL;
7    rGearNumerator           : LREAL;
8    iGearDenumerator         : USINT
9  END_STRUCT
10 END_TYPE
11
```

BECKHOFF

- **fb_MoverCtrl():**
  - **ST_GEAR_DATA**
  - Reference to MasterAxis
    - local var '_Master' must be valid reference
    - Base class contains property to link AXIS_REF of master axis to fb_Mover

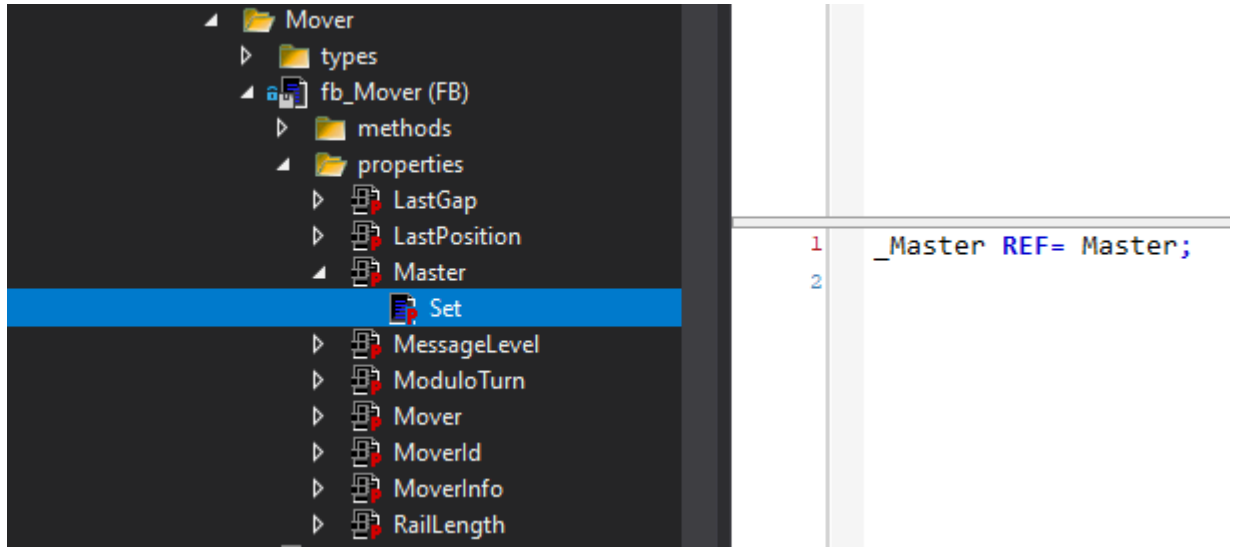
XTS_TRANSPORT | fb_Mover | fb_Mover.Master | ST_GEAR_DATA | E_MOVER_CTRL | ST_MOVE_DATA

```
20  //-----------------------------------------------------------------------------
21  FUNCTION_BLOCK fb_Mover IMPLEMENTS I_XtsTransport_Mover
22  VAR
23      _nMoverId           : UINT;
24
25      _sState             : STRING;
26
27      _Master,
28      _Mover              : REFERENCE TO AXIS_REF;
29      _rLastPosition      : REFERENCE TO LREAL;
30      _rLastGap           : REFERENCE TO LREAL;
31
32      _RailLength         : LREAL;
33      _nSyncStrategy      : Tc3_Mc3Definitions.MC_SYNC_STRATEGY := Tc3_Mc3Definitions.MC
34
35      // mover compact axis information
36      _stInfo             : REFERENCE TO ST_MOVER_INFO; // ActPos, Enable, ...
37
38      _bError             : BOOL;    // used in Cycle()
```

Design

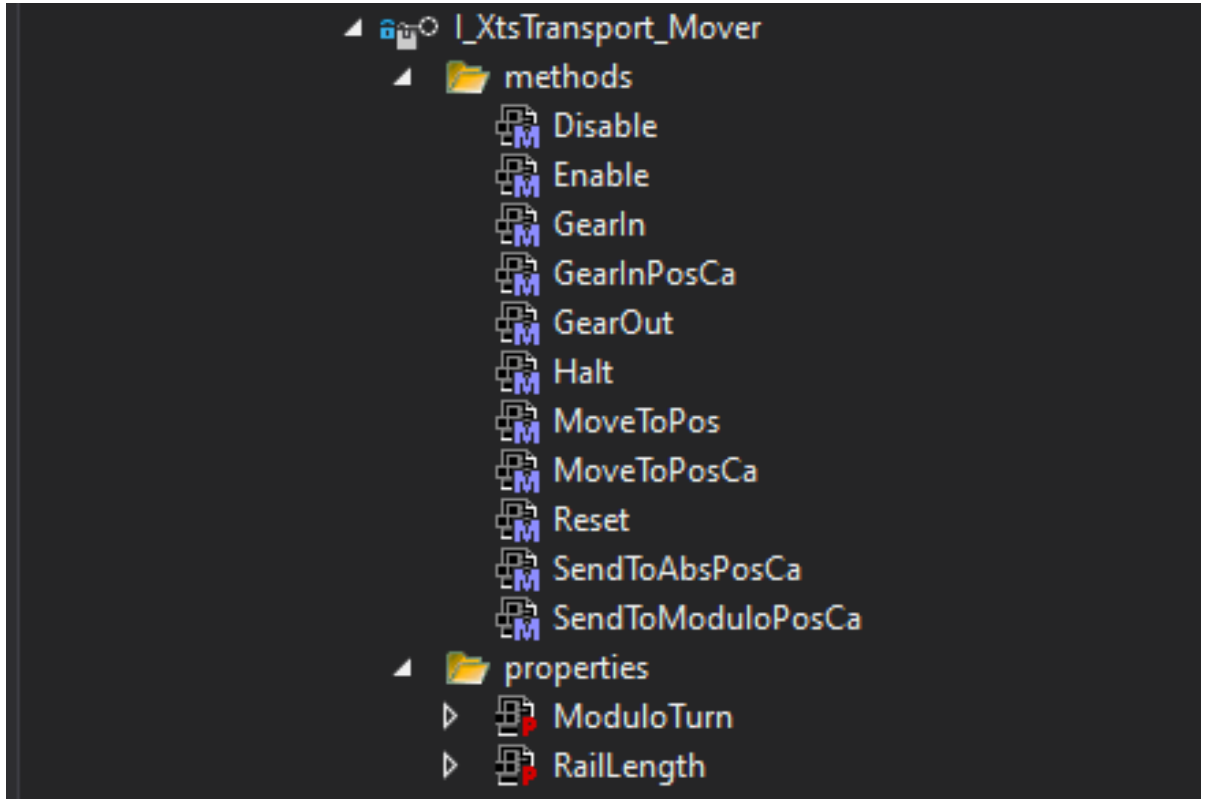
BECKHOFF

- **fb_MoverCtrl():**
  - **ST_GEAR_DATA**
  - Reference to MasterAxis
    - local var ‘_Master’ must be valid reference
    - Base class contains property to link AXIS_REF of master axis to Mover
    - Property you have to set before / on use of gearing motion function



BECKHOFF

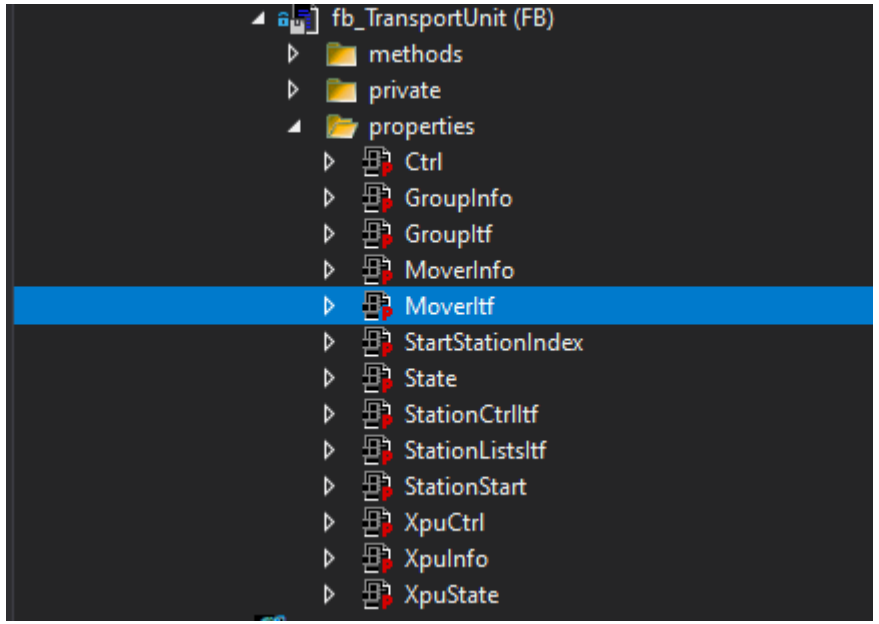
- **I_Transport_Mover:**
  - Interface for use in other classes
    - Fb_TransportUnit
    - Fb_CaGroup
    - Fb_Station
  - Interface is implemented by fb_Mover

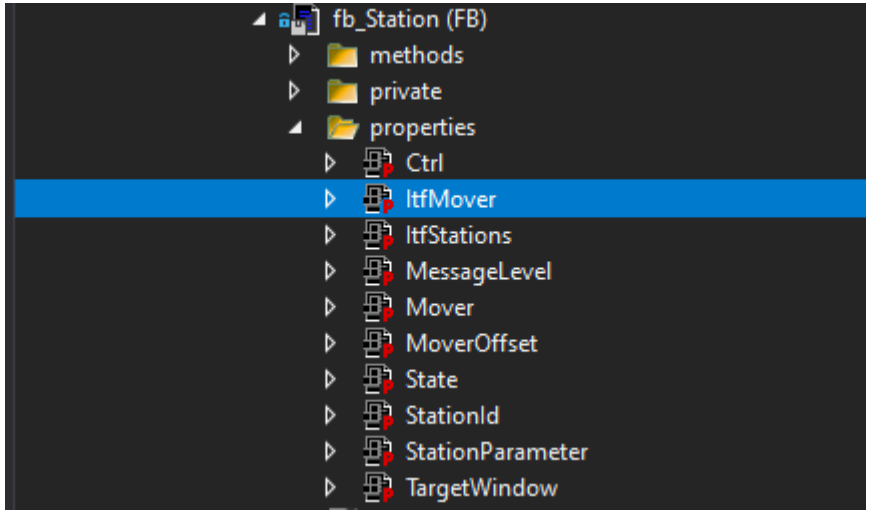


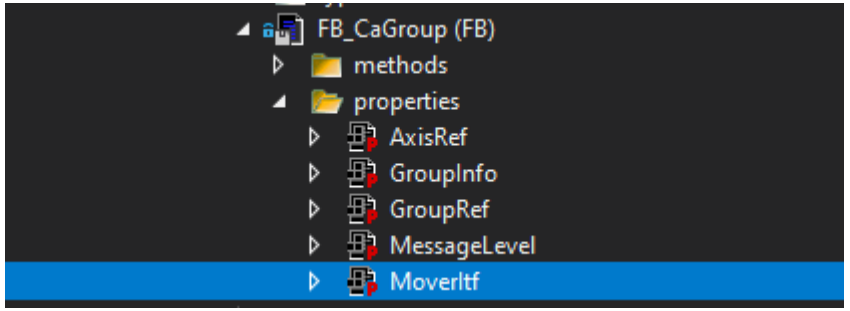
Design

BECKHOFF

- **I_Transport_Mover:**
  - Used in Property by other classes







LICENSE

BECKHOFF

XTS_TRANSPORT_LAYER project