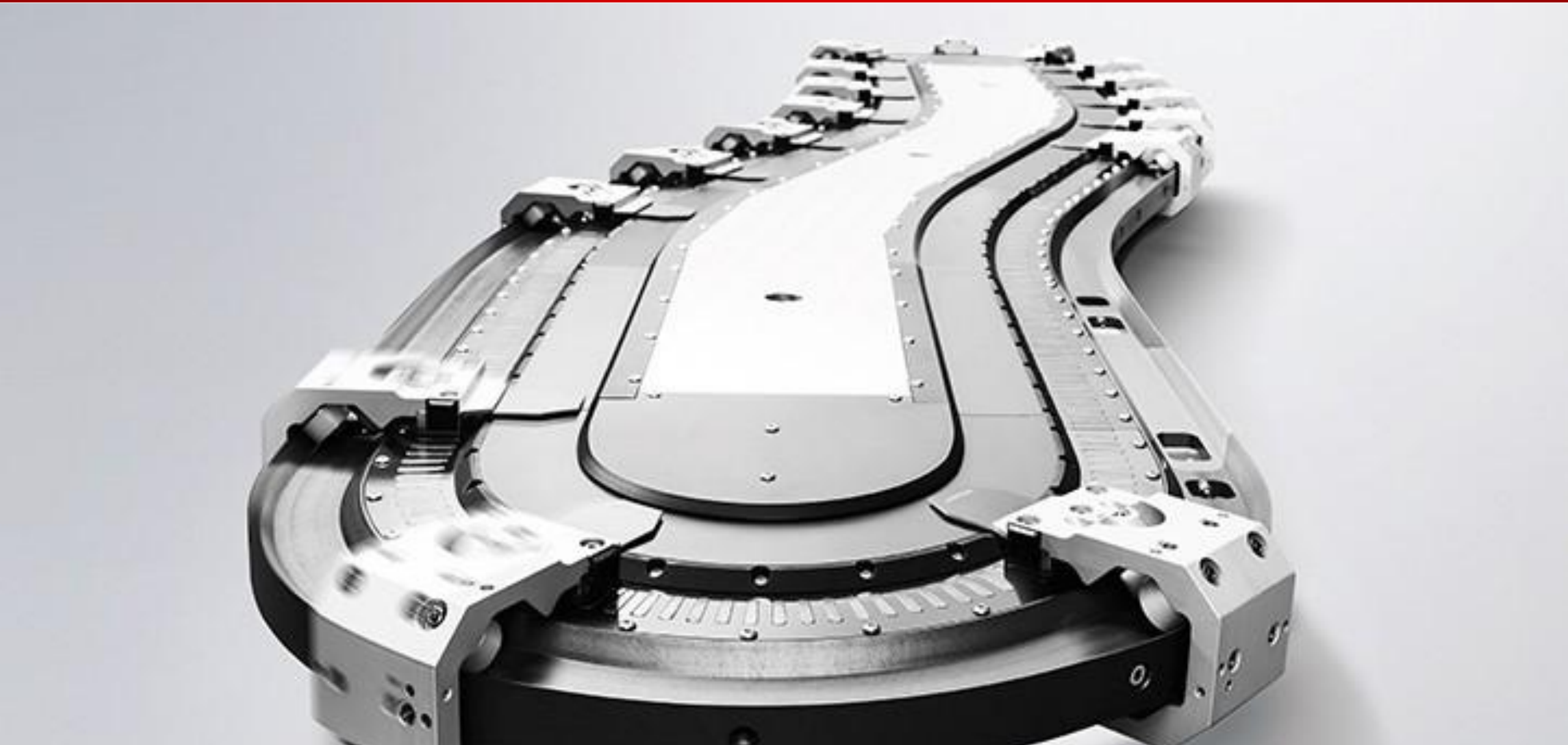
XTS TRANSPORT LAYER – Station Class

BECKHOFF



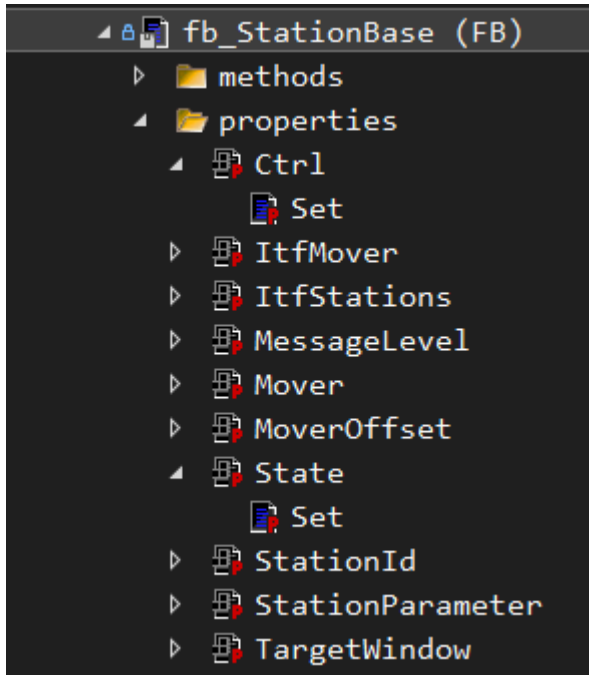
XTS TRANSPORT LAYER – Station Class

BECKHOFF

1. Design
2. Use Cases

BECKHOFF

- **Station based approach**
  - **fb_StationBase**
    - Abstract class
      - Offers uniform station handling
    - Use of REFERENCE pointers
      - Datafields are set via accompanied properties
      - Such properties do not have a Get accessor, since access outside this class shall be done on the original datafield.



Design

BECKHOFF

- **Station based approach**
  - **fb_StationBase**
    - Abstract class
      - Offers uniform station handling
    - Use of REFERENCE pointers
      - Datafields are set via accompanied properties
      - Such properties do not have a Get accessor, since access outside this class shall be done on the original datafield.

```
//#########################################################################
//  XTS Stations
//#########################################################################
FOR nStation := 1 TO MAX_STATION
DO
  IF (GVL_XTS.StationParameter[nStation].eType = E_STATION_TYPE.STATION_PROCESS)
  THEN
    GVL_XTS.Station[nStation].StationId    := nStation;
    GVL_XTS.Station[nStation].MessageLevel:= GVL_MSG.MessageLevelStations[nStation];

    GVL_XTS.StationListItf[nStation]       := GVL_XTS.StationList[nStation];
    GVL_XTS.StationCtrlItf[nStation]       := GVL_XTS.Station[nStation];

    GVL_XTS.Station[nStation].Ctrl         REF= GVL_XTS.StationCtrl;
    GVL_XTS.Station[nStation].State        REF= GVL_XTS.StationState;

    GVL_XTS.Station[nStation].ItfStations REF= GVL_XTS.StationListItf;
    GVL_XTS.Station[nStation].ItfMover     REF= GVL_XTS.MoverItf;
    GVL_XTS.Station[nStation].Mover        REF= GVL_XTS.AxisRefMover;
    GVL_XTS.Station[nStation].MoverOffset REF= GVL_XTS.PositionOffset;


    GVL_XTS.Station[nStation].StationParameter  REF= GVL_XTS.StationParameter;

    // cyclic call
    GVL_XTS.Station[nStation].Cycle();

    // Queue data for each station
    GVL_XTS.StationQueue[nStation]         := GVL_XTS.StationListItf[nStation].Data;
```

Design

BECKHOFF

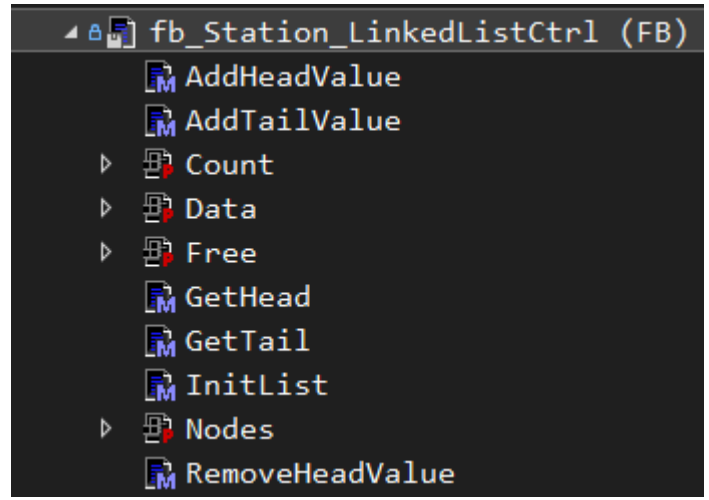
- **Station based approach**
  - **fb_StationBase**
    - Abstract class
      - Offers uniform station handling
    - Use of REFERENCE pointers
      - Datafields are set via accompanied properties
      - Such properties do not have a Get accessor, since access outside this class shall be done on the original datafield.

```
// pointer to all stations
_stCtrl                 : REFERENCE TO ARRAY[1..MAX_STATION]  OF ST_STATION_CTRL;
_stState                : REFERENCE TO ARRAY[1..MAX_STATION]  OF ST_STATION_STATE;

// local copy of command
_eCmd,
_eCmdOld                : E_STATION_CTRL;

_ItfStation             : REFERENCE TO ARRAY[1..MAX_STATION]  OF I_Station_LinkedList;
_ItfMover               : REFERENCE TO ARRAY[1..MAX_MOVER]    OF I_XtsTransport_Mover;

// station related data
_rMoverOffset           : REFERENCE TO ARRAY[1..MAX_STATION]  OF T_NEST_OFFSET;
_stParameter            : REFERENCE TO ARRAY[1..MAX_STATION]  OF ST_STATION_PARAMETER;

// mover axis ref for info
_Mover                  : REFERENCE TO ARRAY[1..MAX_MOVER]    OF AXIS_REF;
```

Design

BECKHOFF

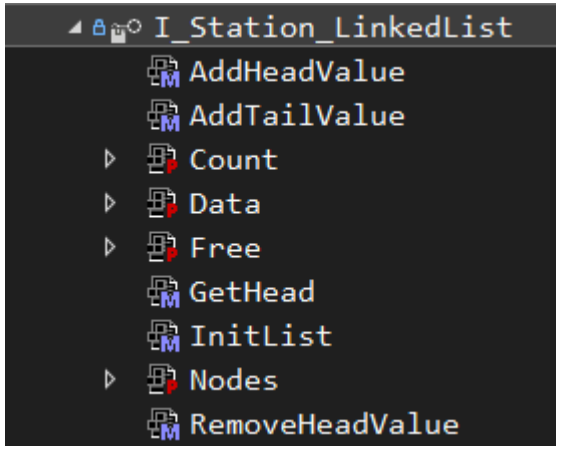
- **Station based approach**
  - **fb_Station_LinkedListCtrl**
    - Linked List
      - Transport of information
        - My ticket.
          - ST_STATION_MOVER_DATA.nMoverId
        - My destination.
          - ST_STATION_MOVER_DATA. nTargetStation
        - My compartement(s).
          - ST_STATION_MOVER_DATA.nMask
        - My seat.
          - ST_STATION_MOVER_DATA.rOffset



Design

BECKHOFF

- **Station based approach**
  - **fb_Station_LinkedListCtrl**
    - **Tc2_Utilities.FB_LinkedListCtrl**
    - Atomic access
    - Global Instances
      - Station Queues for diag and visu
    - Used via Interface




```
// station handshaking with mover and extern process
// station sends mover to target station/WaitPos
// station adds mover data to LinkedList.AddTail() of target station
Station             : ARRAY[1..MAX_STATION] OF fb_StationProcess;
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;

// interface for access to List methods
StationListItf      : ARRAY[1..MAX_STATION] OF I_Station_LinkedList;

// interface for access to station methods
StationCtrlItf      : ARRAY[1..MAX_STATION] OF I_XtsTransport_Station;
```

Design

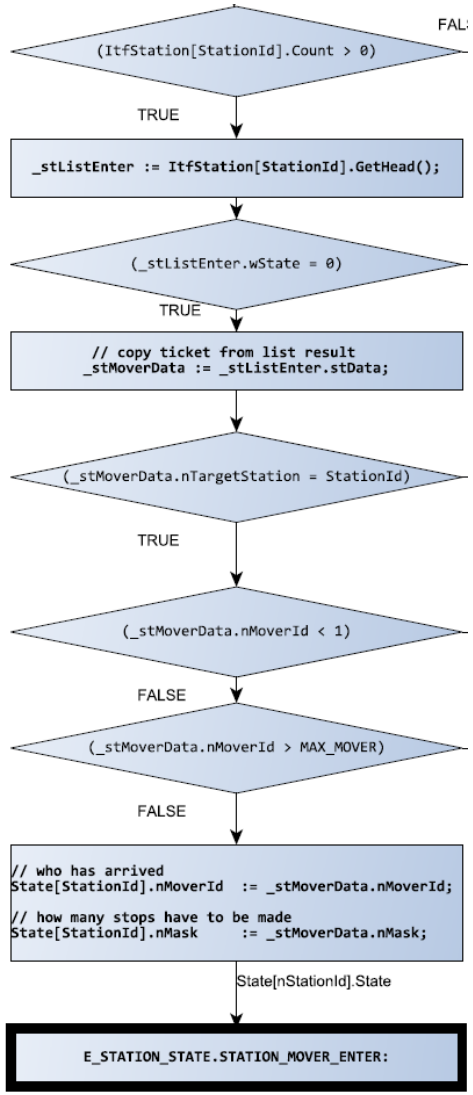
BECKHOFF

- **Station based approach**
  - **fb_StationProcess / fb_StationGearInPos**
    - Extend fb_StationBase
    - Global array of indexed Stations (nStationId)
    - Cycle()
      - State Machine for handshaking movements of mover in station
        - Ctrl/State pair
      - Mover is detected by:
        - _ItfStation[nStationId].Count > 0
        - Mover ID is copied from ticket
          - → LinkedList must be correct!
            - → Movement is used for inherently sorted list.

Design

BECKHOFF

- **Station based approach**
  - **fb_StationProcess / fb_StationGearInPos**
    - Use of LinkedList at station infeed:
      - Get top entry of list (Head)
        - Plausibility checks of ticket data
    - Station switched to **STATION_MOVER_ENTER**
    - You decide what to do next.
      - Disable:
        - **E_STATION_CTRL.STATION_DISABLE**
      - Infeed to position / GearIn to MasterAxis:
        - **E_STATION_CTRL.STATION_MOVER_ENTER**
      - Send mover to new target:
        - **E_STATION_CTRL.STATION_MOVER_SEND**



- **Station based approach**
  - **fb_StationProcess - E_STATION_CTRL.STATION_MOVER_ENTER:**

    - **E_PROGRESS_INIT:**
    - Checks after seeing the command to let mover into station:
      - Get first active nest position in _stMoverData.nMask (1 = default)
        - If nMask == 0 → only one Stop, then mover has to leave
          - For 1 stop stations you need not to use nMask

    - **E_PROGRESS_BUSY:**
    - MoveIn(): prepares movement to PosStop with all offsets included
    - Check whether mover has to cross modulo turn

BECKHOFF

- **Station based approach**
  - **fb_StationGearInPos - E_STATION_CTRL.STATION_MOVER_ENTER:**
    - **E_PROGRESS_INIT:**
      - Same as before
    - **E_PROGRESS_BUSY:**
    - Check for minimal distance to sync position (warning set if not)
    - **E_PROGRESS_PREPARE:**
    - MoveIn(): prepares movement to SlaveSyncPos with all offsets included
    - Check whether mover has to cross modulo turn

BECKHOFF

- **Station based approach**
  - **fb_StationProcess - E_STATION_STATE.STATION_MOVER_IN_TARGET:**

    - Start movement:
      - MoveToPosCA – movement with InTarget and NotMoving check.
        - _Result (E_PROGRESS)
        - Checks for DONE or ERROR and sets state machine accordingly

        - → E_STATION_STATE.STATION_PROCESS_START
          - Handshake state(s) for your process flow
          - See flowcharts for details (Example pdfs)

BECKHOFF

- **Station based approach**
  - **fb_StationGearInPos - E_STATION_STATE.STATION_MOVER_IN_TARGET:**

    - Start GearInPos:
      - GearInPosCA – GearIn to MasterAxis at SlaveSyncPosition.
        - _Result (E_PROGRESS)
        - Checks for DONE or ERROR and sets state machine accordingly
        - PROGRESS_DONE:
          - StartPosition of sync movement is latched
          - Mover is now nSync with the master AND is still moving
            - → E_STATION_STATE.STATION_PROCESS_START
            - Handshake state(s) for your process flow
              - Three options available
                - MOVER_OUT: fast release without checking SyncDistance
                - PROCESS_START: , PROCESS_DONE)

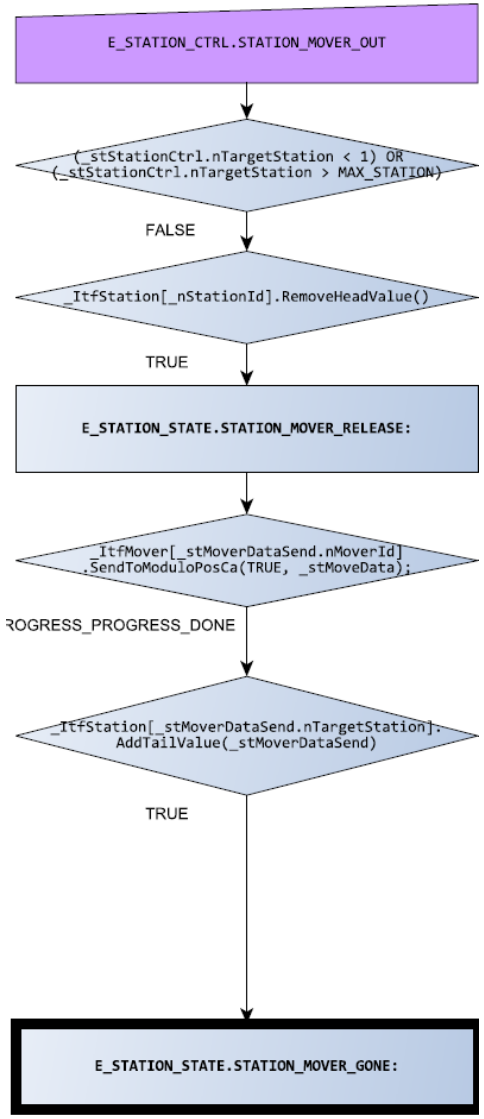
- **Station based approach**
  - **fb_StationGearInPos - E_STATION_STATE.STATION_PROCESS_START:**
    - Mover IS moving nSync with MasterAxis:
      - According to _stParameterGear.eDistance: Synchrone Distance(DiffPosition) calculation starts
      - Handshake state(s) for your process flow
        - Options available for Ctrl:
          - MOVER_OUT: fast release without checking SyncDistance
          - PROCESS_START, PROCESS_DONE:
            - state change to STATION_PROCESS_DONE.

- **Station based approach**
  - **fb_StationGearInPos - E_STATION_STATE.STATION_PROCESS_DONE:**
    - Mover IS still moving nSync with MasterAxis:
      - According to _stParameterGear.eDistance: Synchronous Distance(DiffPosition) calculation continues
      - Handshake state(s) for your process flow
        - Three handshake options available for Ctrl:
          - MOVER_OUT: fast release without checking SyncDistance
          - PROCESS_START: **requires** second handshake
          - PROCESS_DONE: changes to MOVER_OUT after having moved SyncDistance

Design

BECKHOFF

- **Station based approach**
  - **fb_StationProcess / fb_StationGearInPos**
    - Use of LinkedList at station outfeed:
      - Get ticket data from Ctrl(you)
        - Plausibility checks of ticket data
      - Wait for command from Ctrl
        - Delete top entry of LinkedList
      - Wait until mover has moved specified distance
        - ST_STATION_PARAMETER.rReleaseDistance
      - Add bottom (Tail) entry in LinkedList of ST_STATION_CTRL.nTargetStation.



BECKHOFF

- **Station based approach: ST_STATION_PARAMETER**
  - Configuration parameters
    - Station Type
    - Absolute modulo positions as target
      - Waiting Position
    - Relative stop positions
      - Additional quantification of possible stop positions
    - Dynamic constraint of mover in station
    - Relative distance to leave station

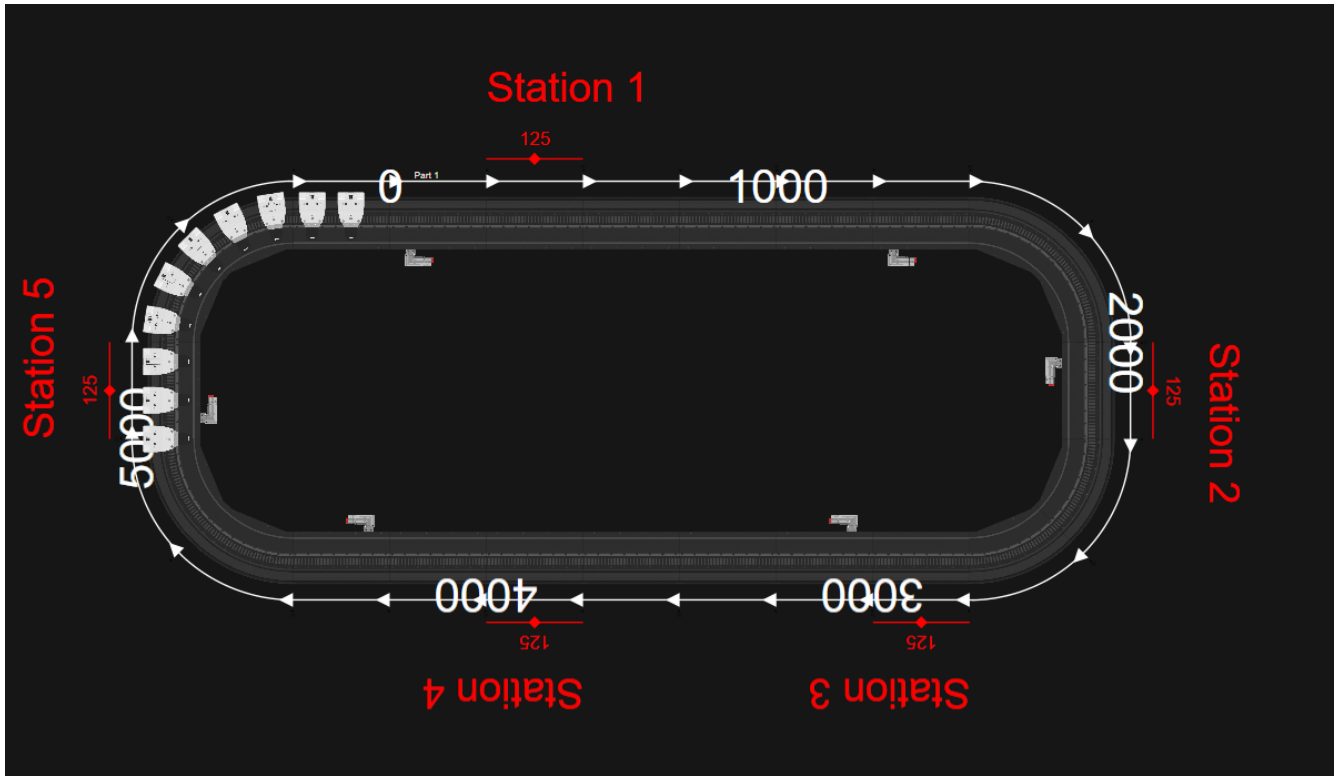
MAIN | ST_STATION_PARAMETER | GVL_XTS

```
TYPE ST_STATION_PARAMETER :
STRUCT
  eType             : E_STATION_TYPE := 1;  // StationProcess or StationGearInPos
  sText             : STRING(80);           // only description
  rPosWait          : REAL;                 // start of station,
                                            // a sending station is using this value
                                            // to send mover to

  rReleaseDistance  : REAL;                 // distance mover has to travel (from ActPos)
                                            // in order for station to go back to disable

  rGap              : REAL;
  rVelo             : REAL;
  rAccDec           : REAL;
  rJerk             : REAL;

  // how many nests (stop positions) mover has to stop at (1 = default)
  nConfiguredStopCount  : USINT := 1; // 1-8 --> NestMask = BYTE

  // mover stop position in station, relative to rPosWait!!
  rPosStop          : ARRAY[1..8] OF LREAL;
END_STRUCT
END_TYPE
```

Use Cases

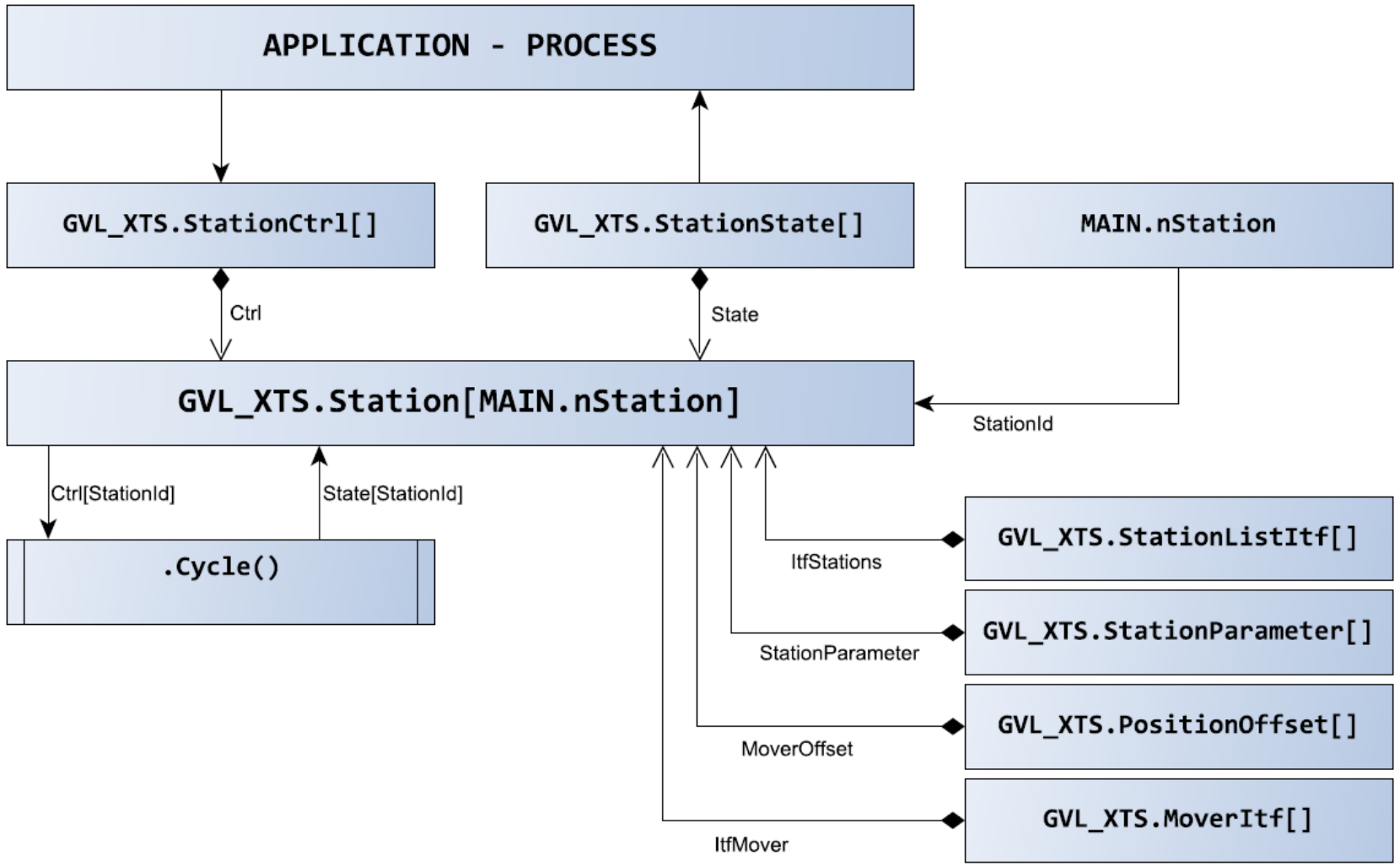
BECKHOFF

- **XTS_DEMO_11**
  - Simple single stations
    - One stop only
    - Target is always next station



BECKHOFF

- **XTS_DEMO_11**
  - Simple handshakes
    - Ctrl/State pair

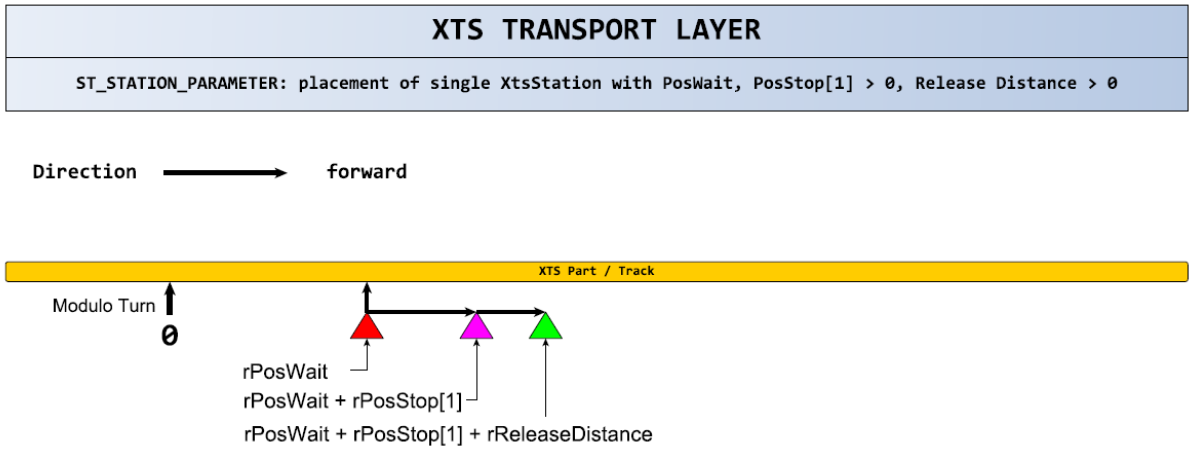


Use Cases

BECKHOFF

- **XTS_DEMO_11**
  - Station configuration
    - WaitPos (absolute modulo)
    - ConfiguredStopCount := 1
    - StopPos[1] (relative)
    - ReleaseDistance > 0

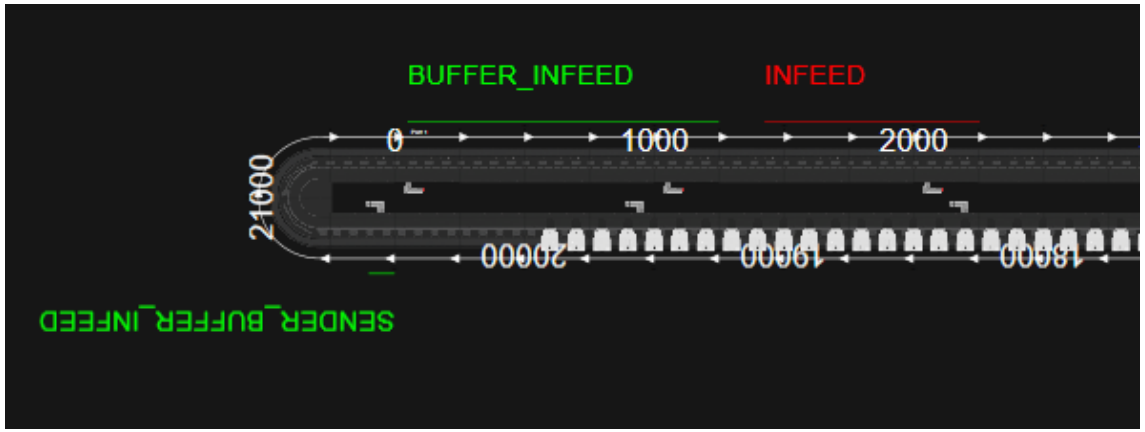
Ex01



Use Cases

BECKHOFF

- XTS_DEMO_APPLICATION_108
  - Transfer system’s parts are fed into INFEED
    - Transfer system’s data is connected to BUFFER_INFEED via LinkedList
  - Transfer system writes bitmask into BUFFER_INFEED List
  - BUFFER_INFEED send movers to INFEED according to bitmask
  - Application requires grouping of stations



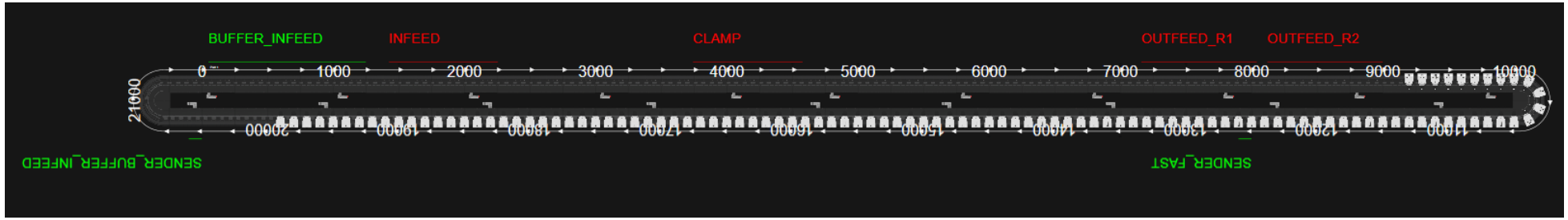
Use Cases

BECKHOFF

- **XTS_DEMO_APPLICATION_108**
  - **Application requires grouping of stations**
    - Process definition:
      - A handshake that may be performed on one or many fb_StationProcess simultaneously
    - Use of global stations Ctrl/State pairs
      - Passed by Reference into process
    - Stations must be mutable (bitmask)
      - Is done before enabling of stations
- **..**
  - Stations work parallel
    - One Ctrl/State pair for process ST_PROCESS_CTRL ST_PROCESS_STATE
  - Range of stations must be defined
    - Closed interval must be strictly monotonically increasing
      - FirstStation (index in global array)
      - LastStation (index in global array)

BECKHOFF

- **XTS_DEMO_APPLICATION_108**

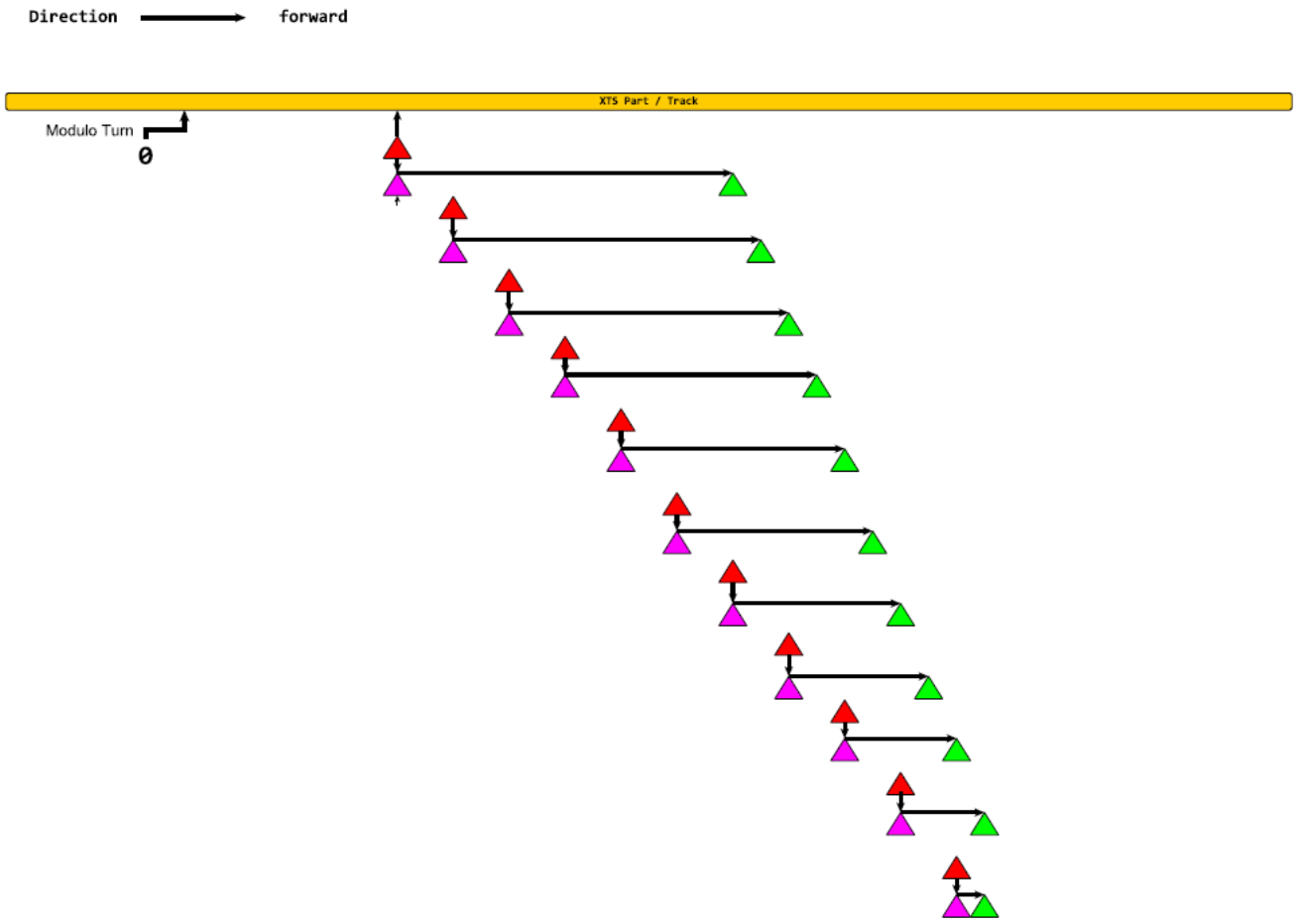


  - **Application requires product transport without empty movers.**
    - BUFFER_INFEED[1] : controls one station
      - Target is INFEED[1 to 12] as specified in its ListEntry (ActivateStation bitmask written by transfer system)
      - Batch from transfer system may contain gaps → mover **must not** be sent to gap.
    - INFEED[1 to12] : controls 12 stations
      - One stop only
      - Target BUFFER_OUTFEED[1]

Use Cases

BECKHOFF

- **XTS_DEMO_APPLICATION_108**
  - See placement example pdfs in doc folder!
  - Station parameters are hard coded in MAIN actions
  - Process parameters are hard coded in MAIN_APP actions
  - For a real production machine, an XML DataServer example is included in the project tree.

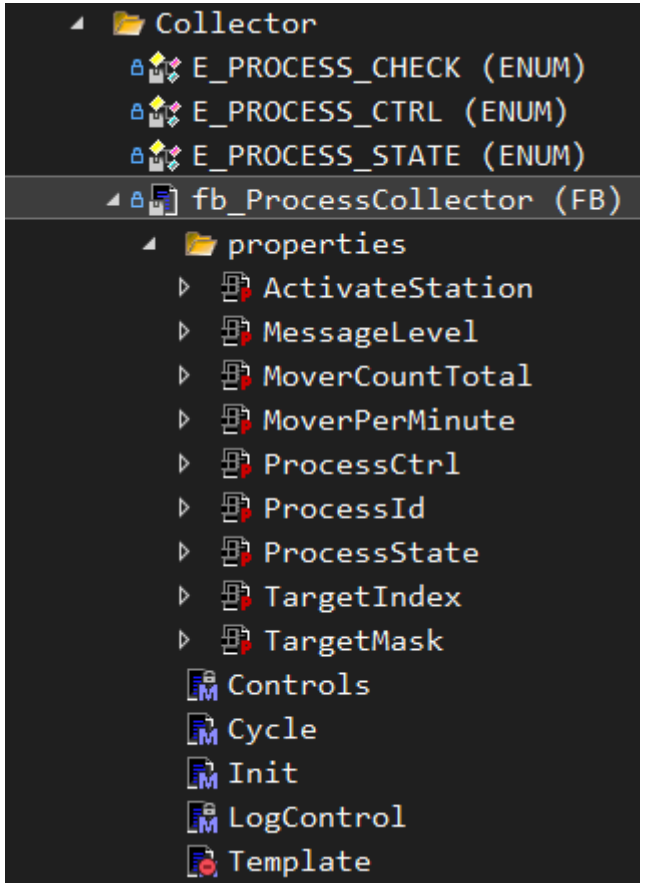


Use Cases

BECKHOFF

- **XTS_DEMO_APPLICATION_108**
  - fb_ProcessCollector
    - Class for grouping stations Ctrl/State pairs
    - Writes commands to stations

```
FUNCTION_BLOCK fb_ProcessCollector EXTENDS fb_StationCollector IMPLEMENTS I_ProcessCollector
VAR
  _nProcessId     : E_INSTANCE; // whoami

  _stControl      : REFERENCE TO ST_PROCESS_CTRL;  // ctrl via property
  _stState        : REFERENCE TO ST_PROCESS_STATE; // state via property

  _eCmd,                           // command for controlling the XtsStations
  _eCmdOld        : E_PROCESS_CTRL;  // logging of command on change
  _eStateProgress : E_PROGRESS;      // progress sub state for process
  _eResult        : E_PROGRESS;      // progress result for methods

  // ctrl words for XtsStations
  {attribute 'displaymode':='bin'}
  _wActivateStation     : T_PROCESS;  // bits enable XtsStations in this process

  // ctrl data for used target XtsStations in target process
  {attribute 'displaymode':='bin'}
  _wTargetMask          : ARRAY[1..SIZEOF(T_PROCESS)*8] OF BYTE;    // mask for multiple PosStop in target
  _rTargetOffset        : ARRAY[1..SIZEOF(T_PROCESS)*8] OF LREAL;   // dyn offset for mulriple PosStop in target

  _nTargetIndex         : ARRAY[1..SIZEOF(T_PROCESS)*8] OF USINT;   // index of XtsStation in target process
```

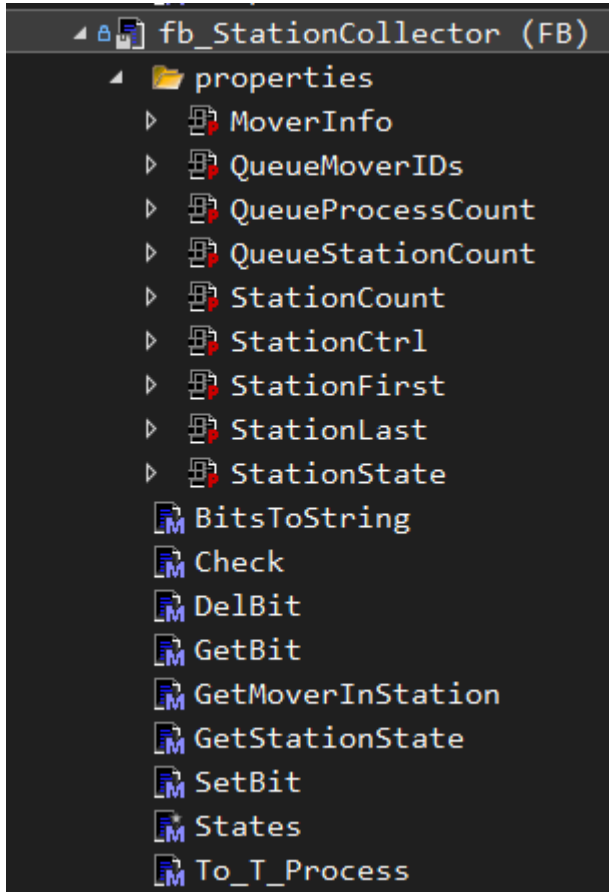


Use Cases

BECKHOFF

- **XTS_DEMO_APPLICATION_108**
  - fb_StationCollector
    - Collects station states in bitmasks

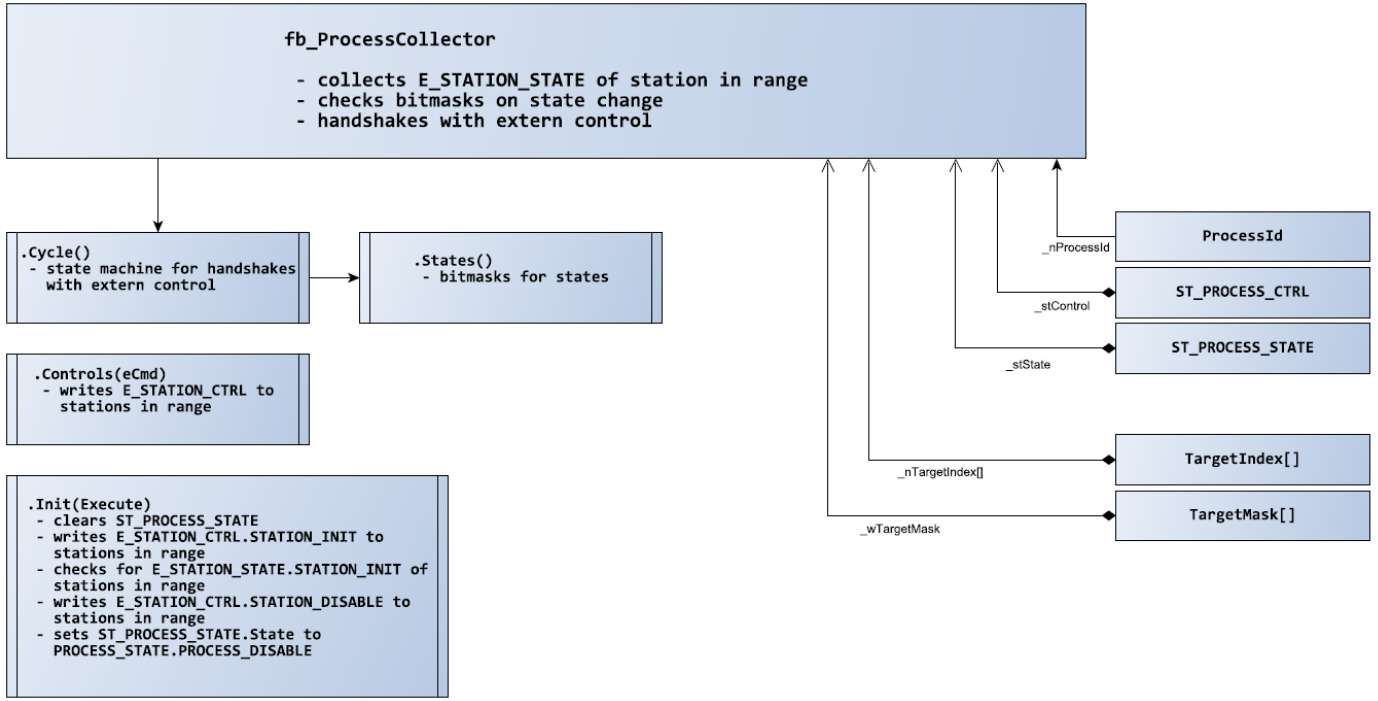
```
FUNCTION_BLOCK fb_StationCollector
VAR
  _eCheck             : E_PROCESS_CHECK;
  _nStationCount      : UINT;

  _nStationFirst,
  _nStationLast       : UINT; // closed range of XtsStations

  // ctrl / state datafields for extern control
  _StationCtrl        : REFERENCE TO ARRAY[1..MAX_STATION] OF ST_STATION_CTRL;
  _StationState       : REFERENCE TO ARRAY[1..MAX_STATION] OF ST_STATION_STATE;

  // mover info datafield
  _MoverInfo          : REFERENCE TO ARRAY[1..MAX_MOVER] OF ST_MOVER_INFO;
```



Use Cases

BECKHOFF

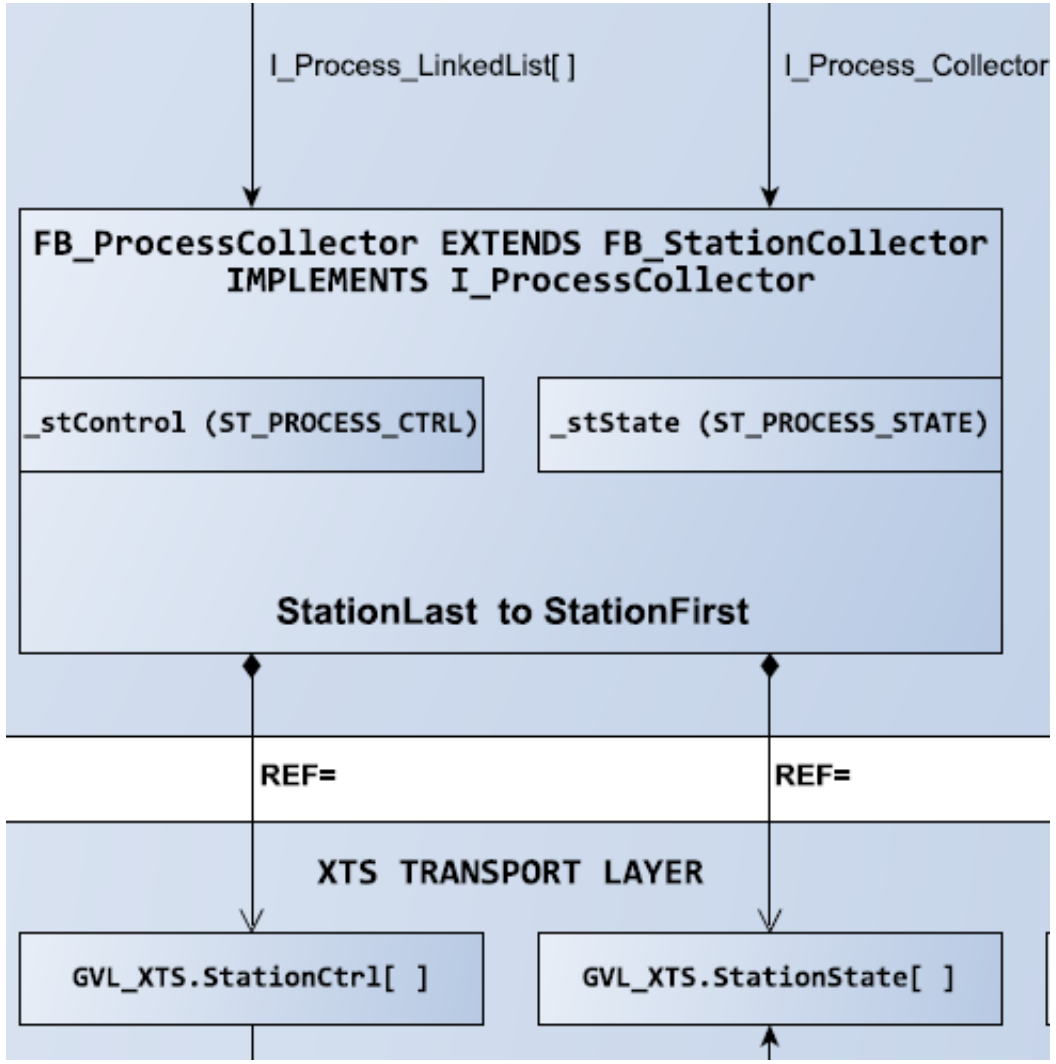
- **XTS_DEMO_APPLICATION_108**
  - Process Ctrl/State pairs
    - Single command structure for grouped stations
    - See handshake fb_ProcessCollector_Cycle.pdf in doc folder of project



Use Cases

BECKHOFF

- **XTS_DEMO_APPLICATION_108**
  - Process Ctrl/State pairs
    - Single command structure for grouped stations
    - See handshake fb_ProcessCollector_Cycle.pdf in doc folder of project



LICENSE

BECKHOFF

XTS_TRANSPORT_LAYER project