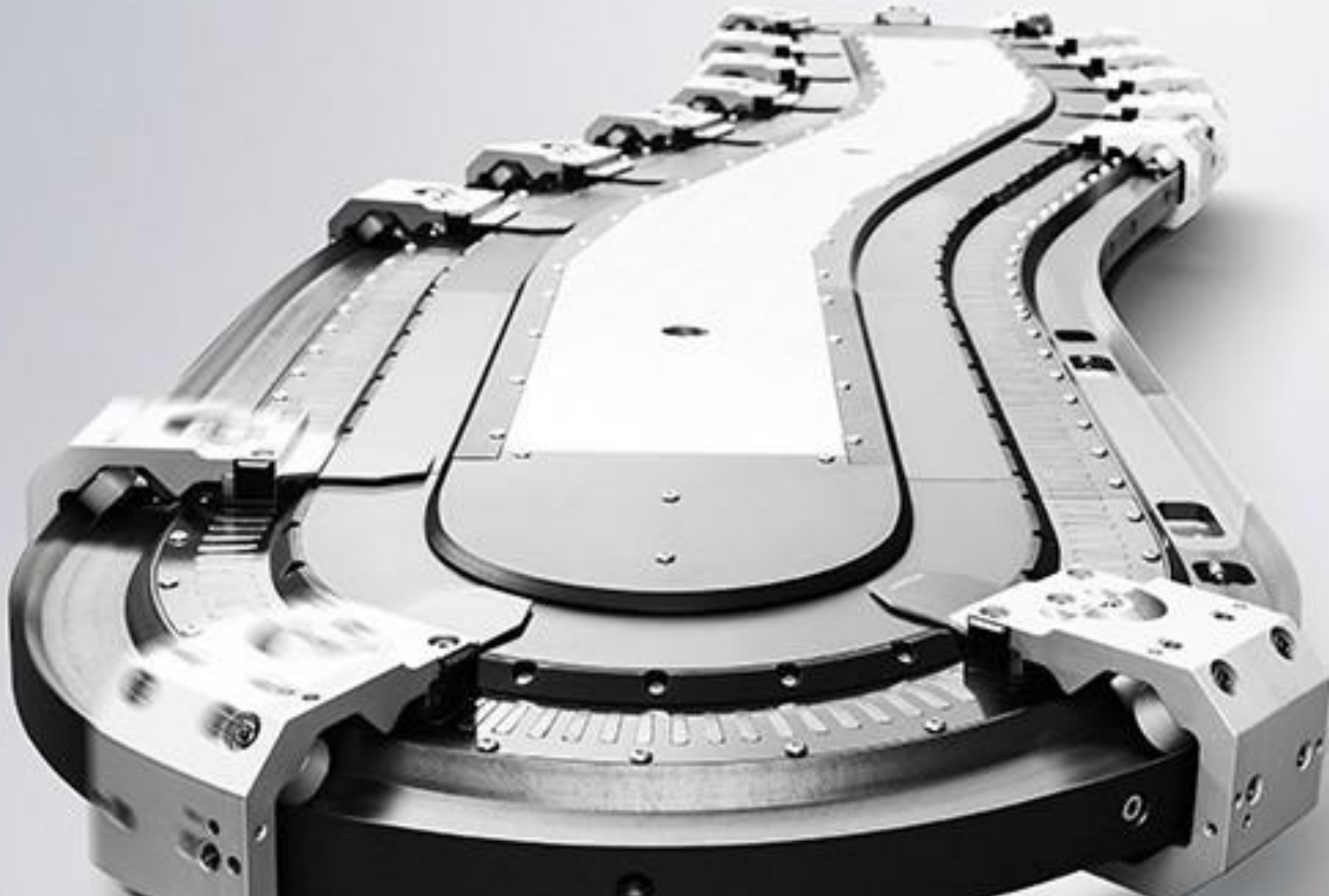


XTS TRANSPORT LAYER – Station Class

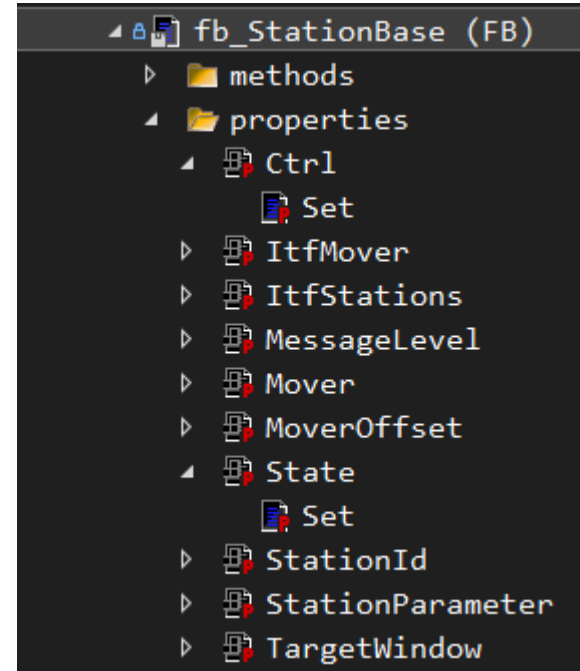
BECKHOFF



1. Design
2. Use Cases



- **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.



- **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;
```

- **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;

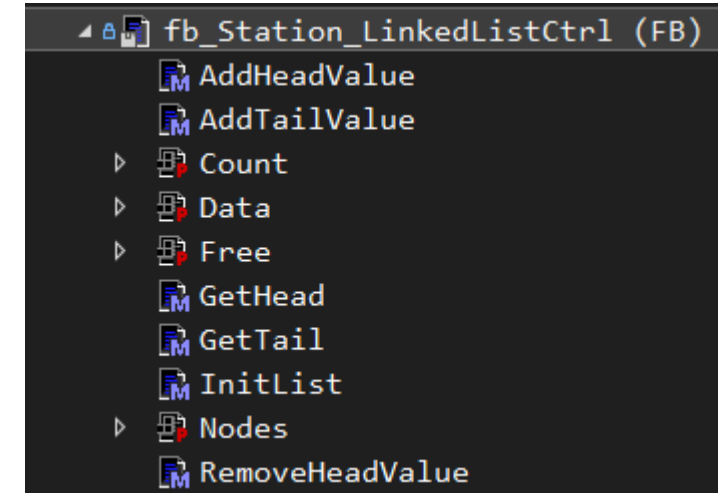
// local copy of command
_eCmd,
_eCmdOld     : E_STATION_CTRL;

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

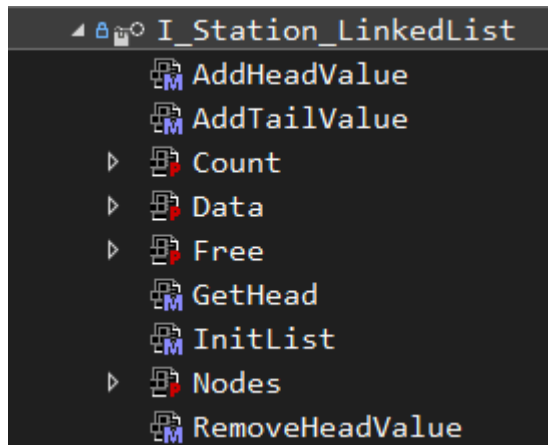
// station related data
_rMoverOffset : REFERENCE TO ARRAY[1..MAX_STATION] OF T_NEST_OFFSET;
_stParameter  : REFERENCE TO ARRAY[1..MAX_STATION] OF ST_STATION_PARAMETER;

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

- **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



- Station based approach
 - fb_Station_LinkedListCtrl
 - Tc2_Uilities.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;

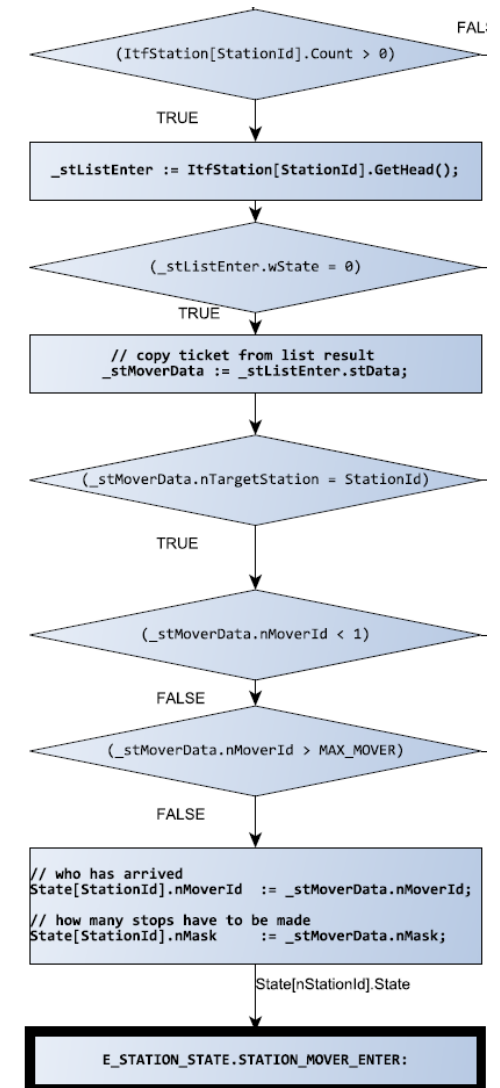
// interface for access to List methods
StationListItf   : ARRAY[1..MAX_STATION] OF I_Station_LinkedList;

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


- **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.

▪ 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

- **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

- **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)

- **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**

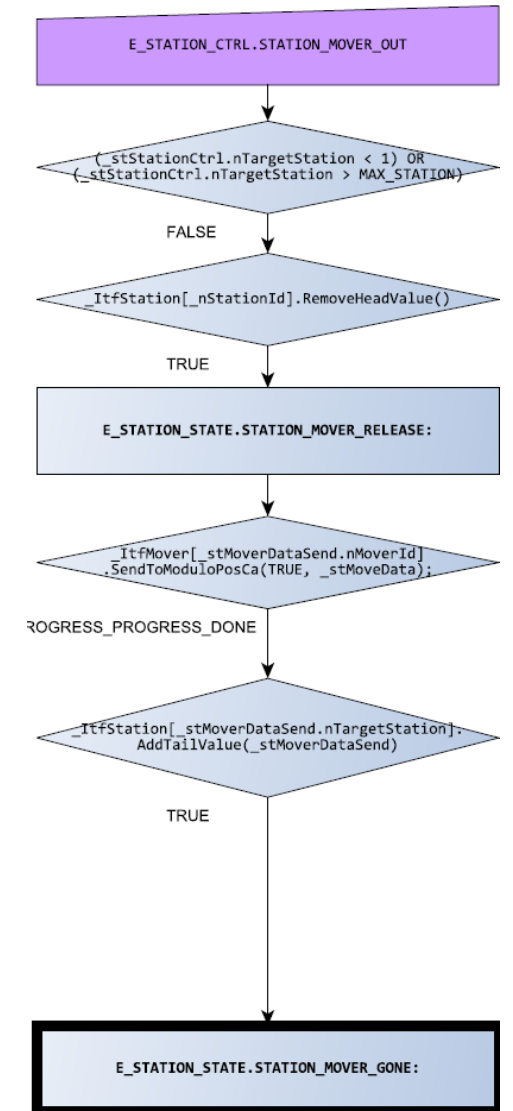
- **fb_StationGearInPos - E_STATION_STATE.STATION_PROCESS_START:**
- Mover IS moving nSync with MasterAxis:
 - According to _stParameterGear.eDistance: Synchrone Distance(DiffPosition) calculation starts
 - Handshake state(s) for your process flow
 - Options available for Ctrl:
 - MOVER_OUT: fast release 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: Synchrone 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

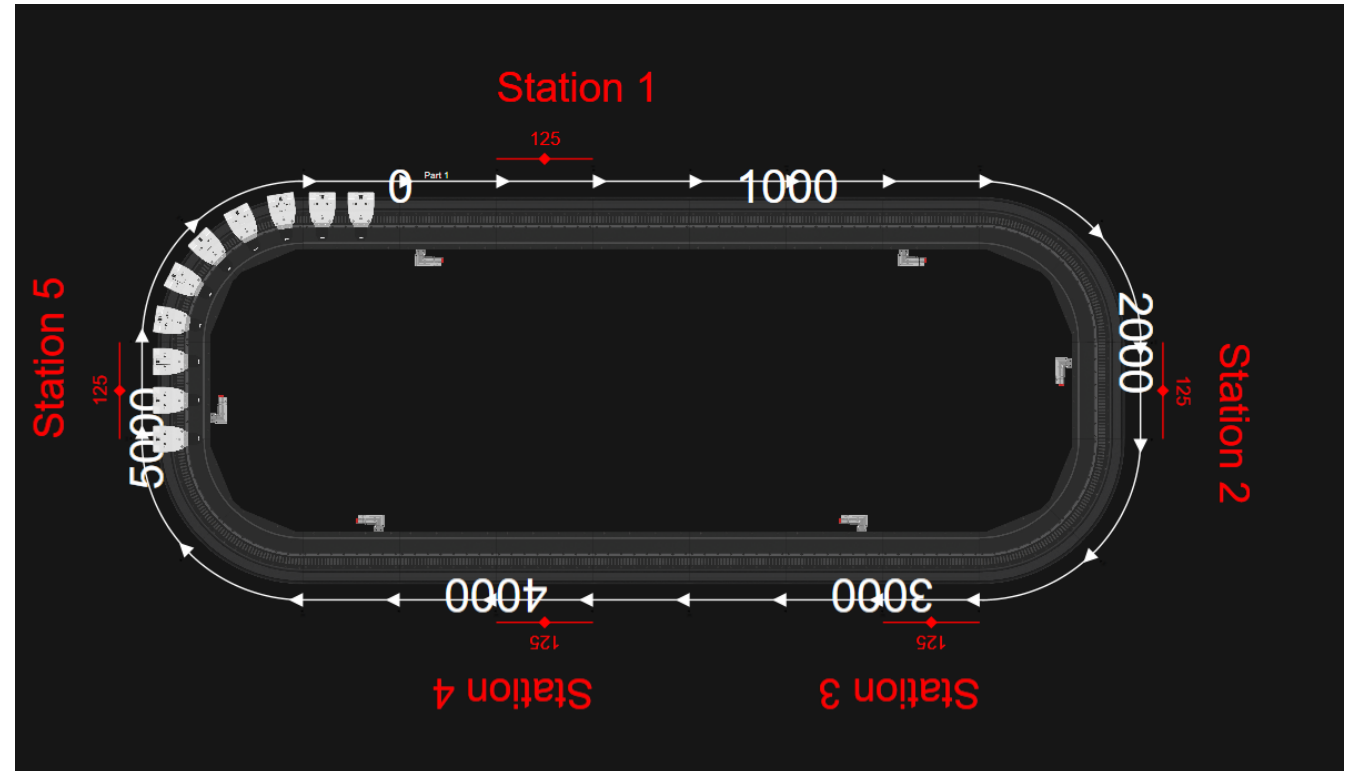
- **Station based approach**
 - **fb_StationProcess / fb_StationGearInPos**
 - Use of LinkedList at station outfeed:
 - Get ticket data from Ctrl
 - Plausibility checks of ticket data
 - Wait for command from Ctrl
 - Delete top entry of LinkedList
 - Wait until mover has moved specified distance
 - ST_STATION_PARAMETER.rReleaseDistance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



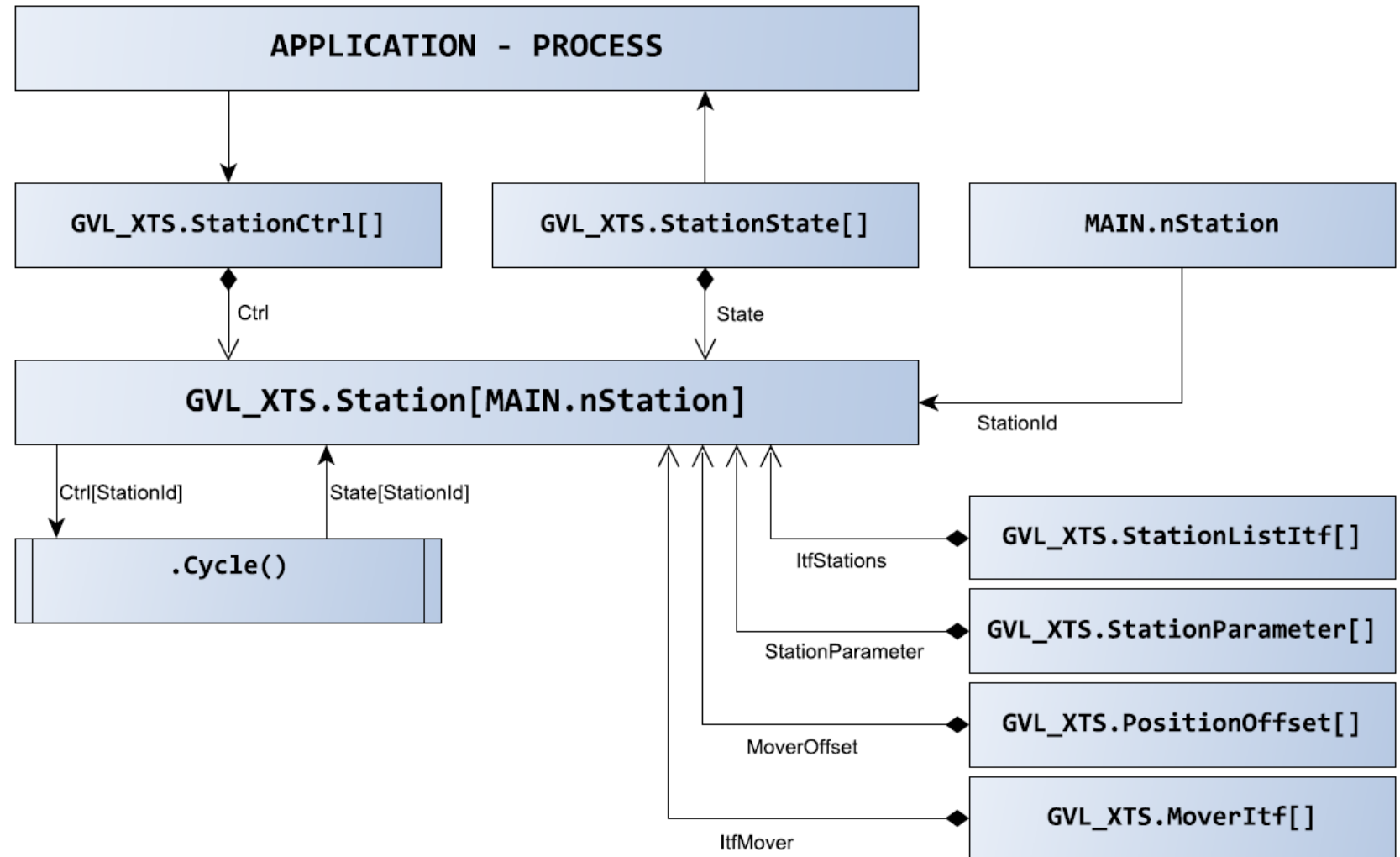
- **Station based approach:**
ST_STATION_PARAMETER
 - Configuration parameters
 - Station Type
 - Absolute modulo positions as target
 - Waiting Position
 - Relative stop positions
 - Additional quantification of possible stop positions
 - Dynamic constraint of mover in station
 - Relative distance to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

```
MAIN  ST_STATION_PARAMETER  GVL_XTS
1  TYPE ST_STATION_PARAMETER :
2  STRUCT
3      eType          : E_STATION_TYPE := 1; // StationProcess or StationGearInPos
4      sText          : STRING(80);        // only description
5      rPosWait       : REAL;              // start of station,
6                                          // a sending station is using this value
7                                          // to send mover to
8
9      rReleaseDistance : REAL;             // distance mover has to travel (from ActPos)
10                                         // in order for station to go back to disable
11
12      rGap           : REAL;
13      rVelo          : REAL;
14      rAccDec        : REAL;
15      rJerk          : REAL;
16
17      // how many nests (stop positions) mover has to stop at (1 = default)
18      nConfiguredStopCount : USINT := 1; // 1-8 --> NestMask = BYTE
19
20      // mover stop position in station, relative to rPosWait!!
21      rPosStop         : ARRAY[1..8] OF LREAL;
22  END_STRUCT
23  END_TYPE
24
```

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



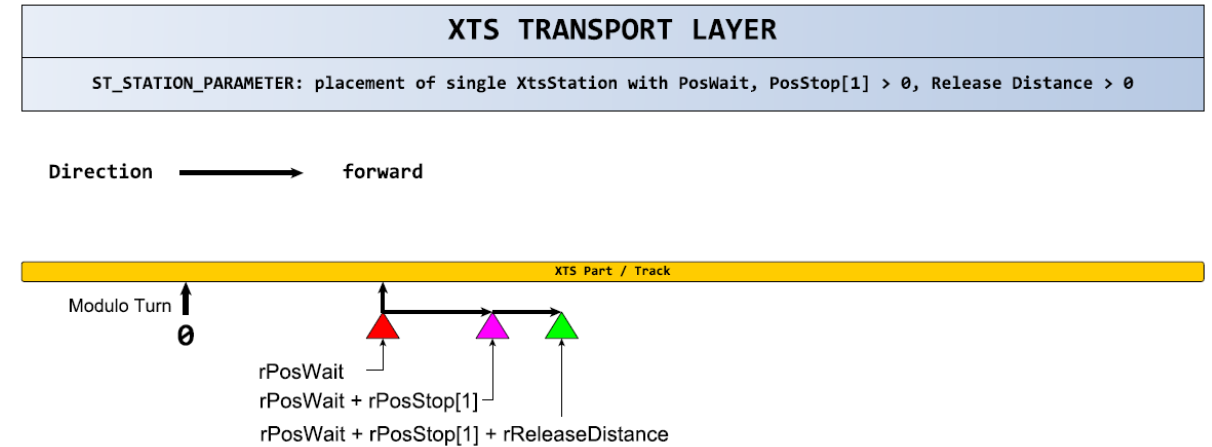
- **XTS_DEMO_11**
 - Simple handshakes
 - Ctrl/State pair



▪ XTS_DEMO_11

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

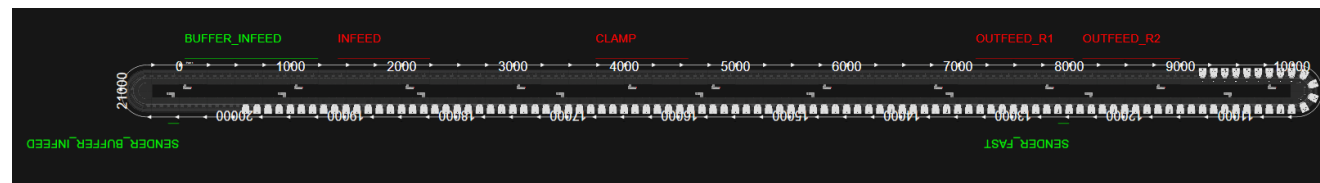
Ex01



- **XTS_DEMO_APPLICATION_108**
 - **Application requires grouping of stations**
 - Process definition:
 - One handshake that may be performed on one or many fb_StationProcess simultaneously
 - Use of global stations Ctrl/State pairs
 - Stations must be mutable
 - Is done before enabling of stations
- ..
 - Stations work parallel
 - One Ctrl/State pair for process
 - Range of stations must be defined
 - Close range
 - LastStation (index in global array)
 - FirstStation (index in global array)

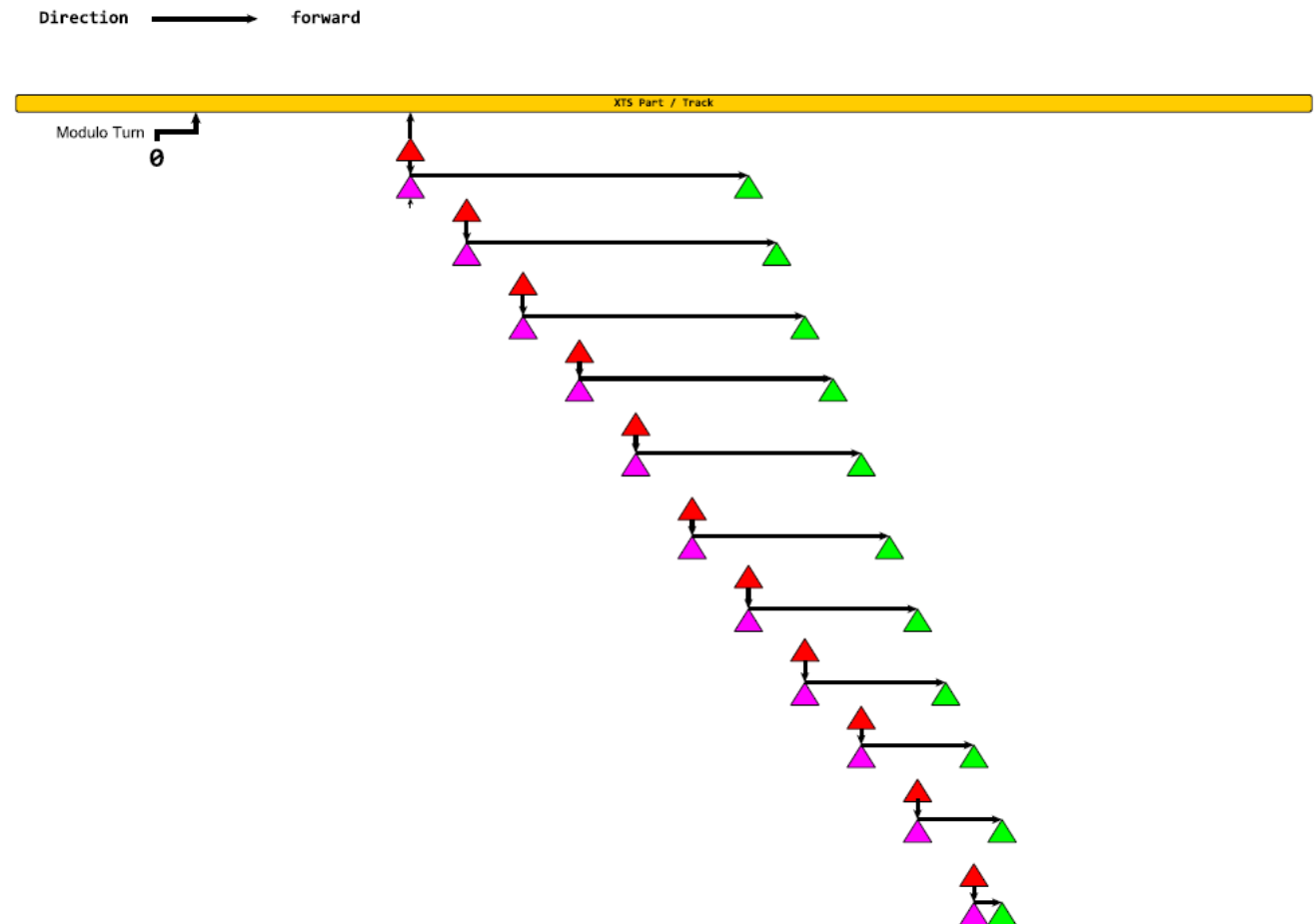
■ XTS_DEMO_APPLICATION_108

- Application requires product transport without gaps.
- Infeed Buffer [1] : controls one station
 - Target Infeed [1 to 12] as specified
 - may contain gaps → no mover must be sent to gap.
- Infeed [1 to 12] : controls 12 stations
 - One stop only
 - Target Outfeed Buffer [1]



▪ 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



▪ 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

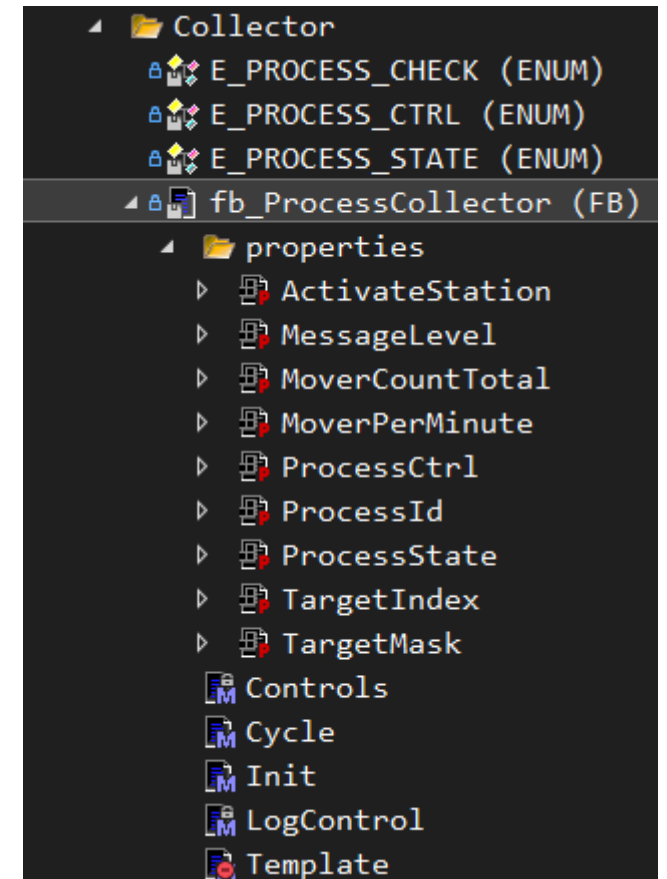
    _stControl        : REFERENCE TO ST_PROCESS_CTRL; // ctrl via property
    _stState          : REFERENCE TO ST_PROCESS_STATE; // state via property

    _eCmd,
    _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
```



▪ XTS_DEMO_APPLICATION_108

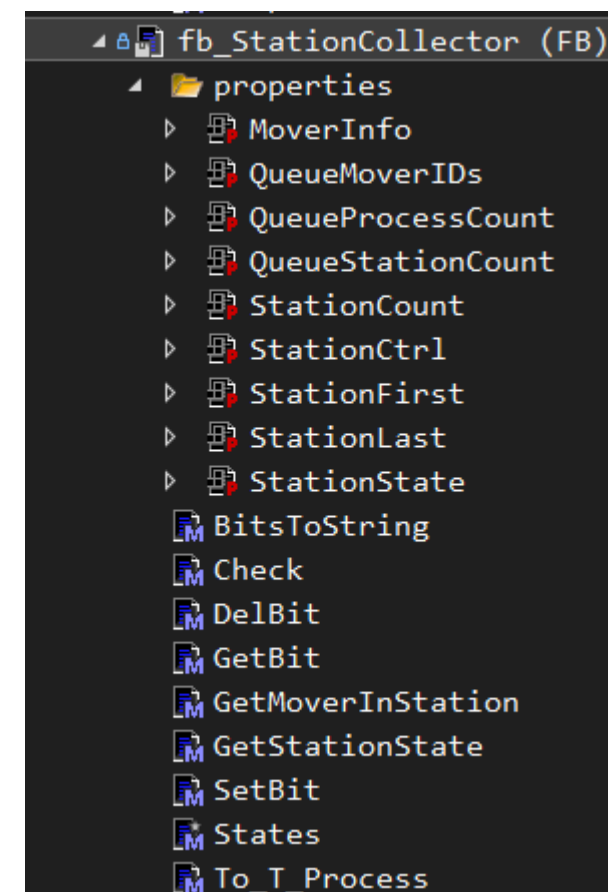
- fb_StationCollector
 - Collects station states in bitmasks

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

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

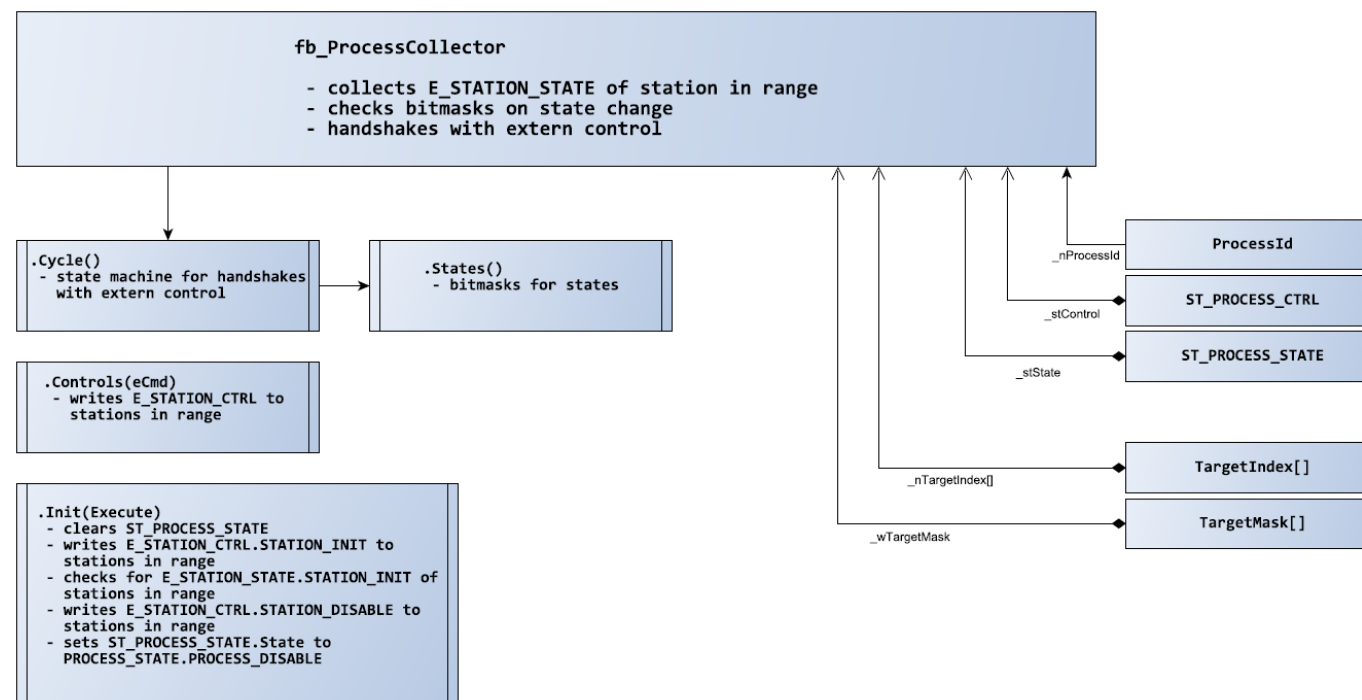
    // ctrl / state datafields for extern control
    _StationCtrl      : REFERENCE TO ARRAY[1..MAX_STATION] OF ST_STATION_CTRL;
    _StationState     : REFERENCE TO ARRAY[1..MAX_STATION] OF ST_STATION_STATE;

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



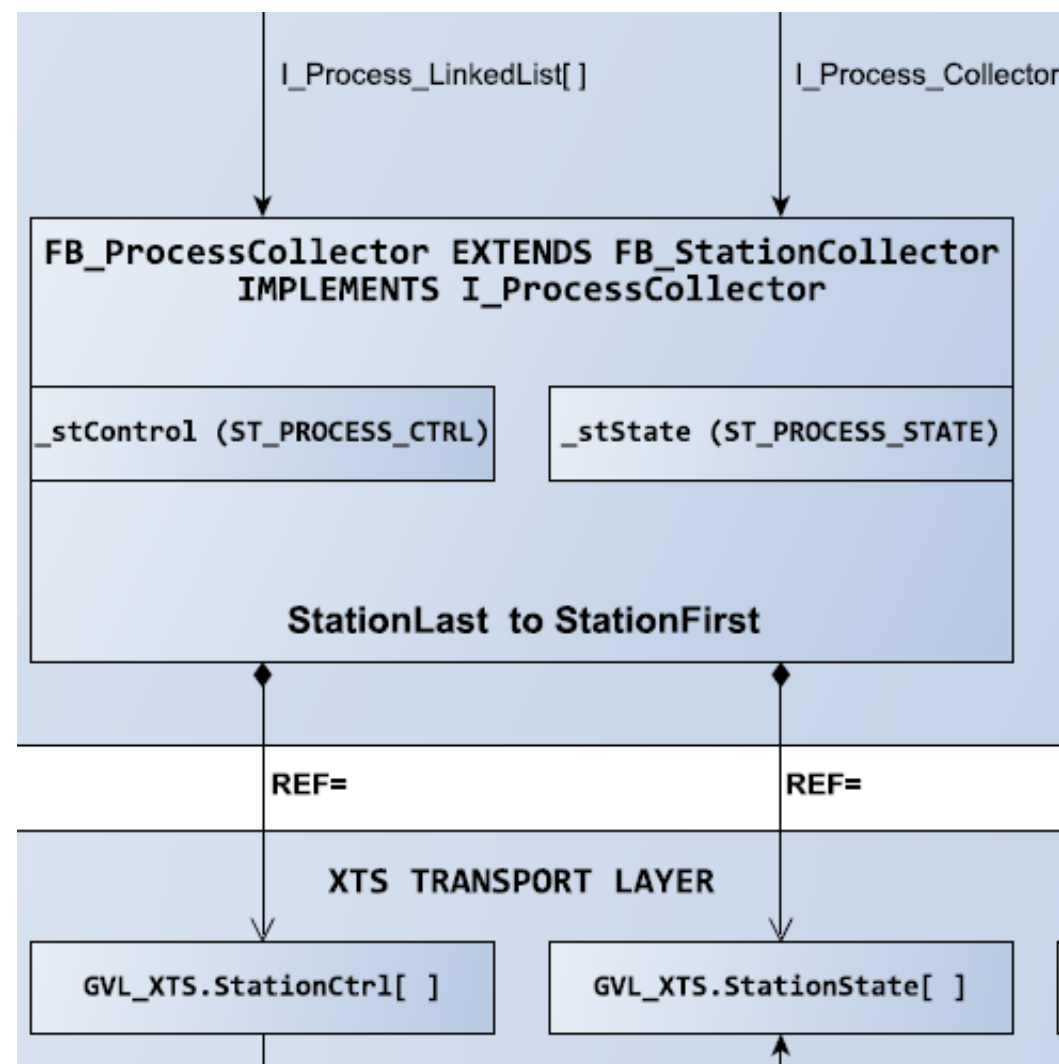
■ XTS_DEMO_APPLICATION_108

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



▪ 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



XTS_TRANSPORT_LAYER project

MIT License

Copyright (c) 2025 HAUD

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.