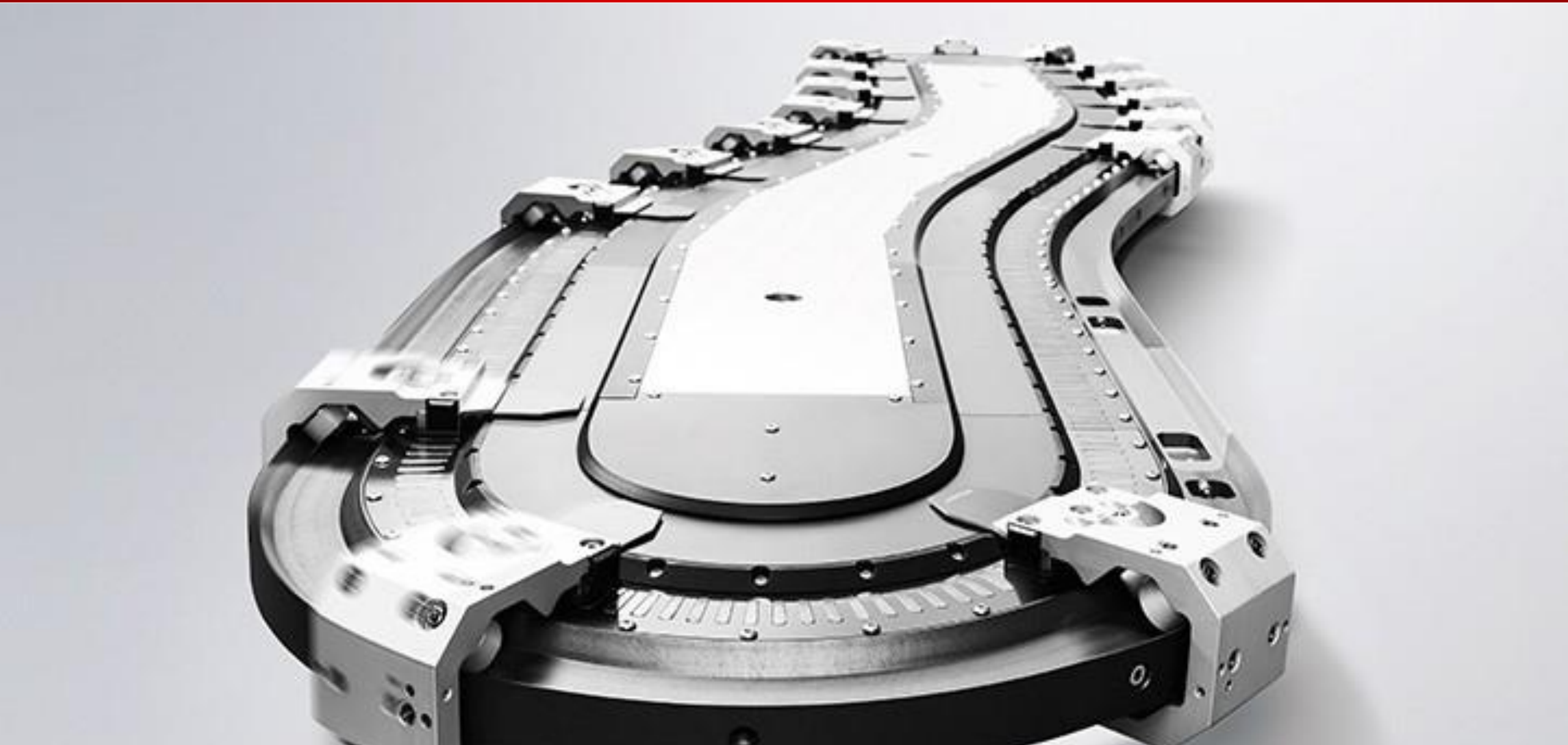
XTS TRANSPORT LAYER – Control and State structures for mapping

BECKHOFF



XTS TRANSPORT LAYER – Fieldbus Mapping

BECKHOFF

1. Introduction
2. Control / State Members
3. Implementation
    1. Profinet
    2. EAP

Introduction

BECKHOFF

- XTS may work as subsystem in heterogen networks
  - Cyclic fieldbus networks
  - NonCyclic TCP/IP networks

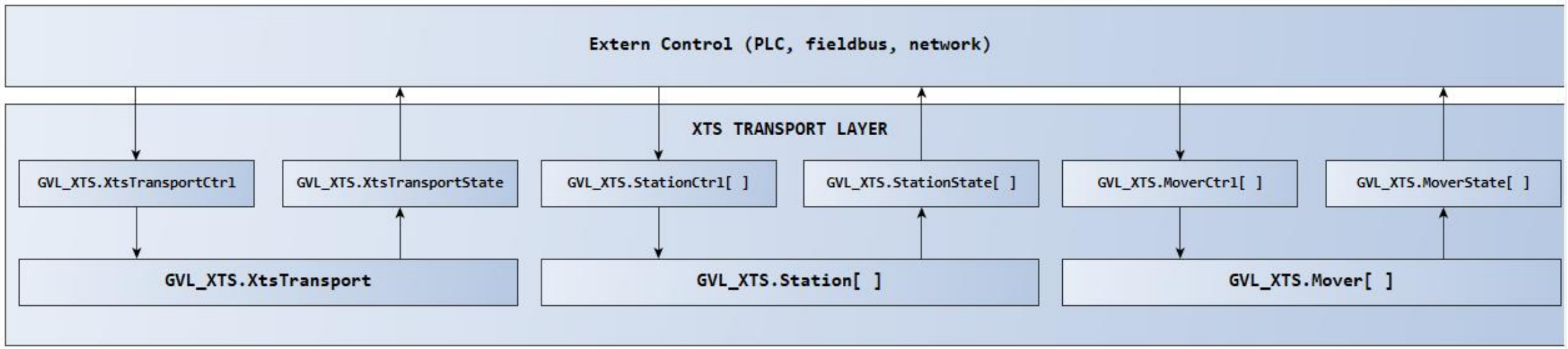
- TwinCAT offers different protocols, the most common (used in this context) are:
  - EtherCAT
  - Profinet
  - EAP

- XTS_TRANSPORT_LAYER must be modified to match your use case
  - Size of mapping must be taken into consideration
  - Cycle time of mapping may be different to PLC cycle time

Control / State Members

BECKHOFF

- **Top Level structures:**



Control / State Members

BECKHOFF

- **XtsTransport structures:**
  - ST_XTS_TRANSPORT_CTRL
    used for getting XTS_TRANSPORT_LAYER
    to a defined state.

E_XTS_TRANSPORT_CTRL

```
{attribute 'strict'}
{attribute 'to_string'}
TYPE E_XTS_TRANSPORT_CTRL :
(
  CMD_NULL,
  CMD_INIT                  := 10,
  CMD_IDLE,

  CMD_MOVER_ENABLE          := 20,
  CMD_MOVER_DISABLE,
  CMD_MOVER_HALT_CA,
  CMD_MOVER_STOP,

  CMD_GROUP_CLEAR           := 30,
  CMD_GROUP_BUILD,
  CMD_GROUP_ENABLE,
  CMD_GROUP_STOP,

  CMD_TRANSPORT_START       := 40,
  CMD_TRANSPORT_RESTART

)UINT;
END_TYPE
```

ST_XTS_TRANSPORT_CTRL

```
{attribute 'pack_mode' := '2'}
TYPE ST_XTS_TRANSPORT_CTRL :
STRUCT
  Cmd       : E_XTS_TRANSPORT_CTRL;
END_STRUCT
END_TYPE
```

Control / State Members

BECKHOFF

- **XtsTransport structures:**
  - ST_XTS_TRANSPORT_STATE
  - State delivers information about current Command, XPU and CAGroup.
  - State feedback is a combined value of the active command and E_PROGRESS
  - Check feedback for internal checks
  - XpuState feedback is a combined value of the current xpu command and E_PROGRESS
  - See “fb_TransportUnit.pdf” in doc folder of this project.

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
```

Control / State Members

BECKHOFF

- **XtsTransport structures:**
  - E_XTS_TRANSPORT_STATE:
    - Enum feedback for ctrl
    - Combined with E_PROGRESS

E_PROGRESS

```
{attribute 'qualified_only'}
//{attribute 'strict'}
{attribute 'to_string'}
TYPE E_PROGRESS :
(
  // progress is used in project to
  // mirror state of requested command/function
  PROGRESS_INVALID,
  PROGRESS_NOT_EXIST      :=   100,
  PROGRESS_INIT           :=  1000,
  PROGRESS_BUSY           :=  2000,
  PROGRESS_PREPARE        :=  3000,
  PROGRESS_STARTUP        :=  4000,
  PROGRESS_CHECK          :=  5000,
  PROGRESS_OCCUPIED       :=  6000,
  PROGRESS_WORKING        :=  7000,
  PROGRESS_STILL_WORKING  :=  8000,
  PROGRESS_ERROR          :=  9000,
  PROGRESS_DONE           := 10000
)UINT;
END_TYPE
```

E_XTS_TRANSPORT_STATE

```
{attribute 'to_string'}
TYPE E_XTS_TRANSPORT_STATE :
(
  TRANSPORT_NULL,
  TRANSPORT_INVALID,
  TRANSPORT_INIT            := 10,
  TRANSPORT_IDLE,

  TRANSPORT_MOVER_ENABLE    := 20,
  TRANSPORT_MOVER_DISABLE,
  TRANSPORT_MOVER_HALT_CA,
  TRANSPORT_MOVER_STOP,

  TRANSPORT_GROUP_CLEAR     := 30,
  TRANSPORT_GROUP_BUILD,
  TRANSPORT_GROUP_ENABLE,
  TRANSPORT_GROUP_STOP,

  TRANSPORT_START           := 40,
  TRANSPORT_RESTART
)UINT;
END_TYPE
```

Control / State Members

BECKHOFF

- **XtsTransport structures:**
  - E_XTS_TRANSPORT_CHECK:
    - Enum feedback for pointer checks
    - If no pointer error is detected, Check is set to DONE_CHECK.

E_XTS_TRANSPORT_CHECK

```
{attribute 'qualified_only'}
{attribute 'to_string'}
TYPE E_XTS_TRANSPORT_CHECK :
(
  // info for cyclic checks
  INIT_CHECK,
  START_CHECK,    // check cycle start
  XPU_INIT,       // Xpu: initialization


  // errors in fb_TransportUnit.Check()
  POINTER_CHECK           := 900,
  POINTER_NULL_CTRL,

  POINTER_NULL_STATE,
  POINTER_NULL_XPU_CTRL,

  POINTER_NULL_XPU_STATE,
  POINTER_NULL_XPU_INFO,

  POINTER_NULL_GROUP_ITF,
  POINTER_NULL_GROUP_INFO,

  POINTER_NULL_MOVER,
  POINTER_NULL_MOVER_ITF,

  POINTER_NULL_MOVER_INFO,
  POINTER_NULL_MOVER_LAST_POS,
  POINTER_NULL_MOVER_LAST_GAP,

  POINTER_NULL_STATION_START,
  POINTER_NULL_STATION_ITF,
  POINTER_NULL_STATION_CONTROL,
  POINTER_NULL_STATION_STATE,
  POINTER_NULL_STATION_CTRL_ITF,

  DONE_CHECK              := 10000
)UINT;
END_TYPE
```

Control / State Members

BECKHOFF

- **XtsTransport structures:**
  - ST_XPU_STATE:
    - Initialization state, (should be checked on startup)
    - State feedback combined with E_PROGRESS
    - Check feedback for cyclic plausibility checks
    - DcLink 48V feedback from **all** motor modules.

ST_XPU_STATE

```
1  {attribute 'pack_mode' := '2'}
2  TYPE ST_XPU_STATE :
3  STRUCT
4    Init              : E_XPU_INIT;
5    State             : E_XPU_STATE;
6    Check             : E_XPU_CHECK;
7    DcLink            : BIT;
8  END_STRUCT
9  END_TYPE
10
```

Control / State Members

BECKHOFF

- **XtsTransport structures:**
  - E_XPU_INIT:
    - Initialization state, (should be checked on startup)

E_XPU_INIT

```
{attribute 'to_string'}
TYPE E_XPU_INIT :
(
  INIT_START,
  INIT_ENVIRONMENT_OID            := 10,
  INIT_INFO_SERVER_ITF,
  INIT_INFO_SERVER_STATION_COUNT,
  INIT_INFO_SERVER_STATION_OID,
  INIT_CA_GROUP_OID               := 20,
  INIT_CA_GROUP_GET_OID,
  INIT_CA_GROUP_CHECK_OID,
  INIT_CA_GROUP_COM,
  INIT_PARAMETERSET_COUNT,
  INIT_PARAMETERSET_OID,
  INIT_PARAMETERSET_COM,
  INIT_PARAMETERSET_COM_NEXT,
  INIT_PROCESSING_UNIT_COM        := 30,
  INIT_PROCESSING_UNIT_PART_COUNT,
  INIT_PROCESSING_UNIT_PART_OID,
  INIT_PROCESSING_UNIT_TRACK_COUNT,
  INIT_PROCESSING_UNIT_TRACK_OID,
  INIT_PROCESSING_UNIT_MOVER_COUNT,
  INIT_PROCESSING_UNIT_MOVER_OID,
  INIT_PROCESSING_UNIT_TASK_COUNT,
  INIT_PROCESSING_UNIT_TASK_OID,
  INIT_TRACK_COM                  := 50,
  INIT_PART_COM                   := 60,
  INIT_PART_MODULE_COUNT,
  INIT_PART_MODULE_OID,
  INIT_PART_MODULE_COM,
  INIT_PART_MODULE_COM_NEXT,
  INIT_MOVER_COM                  := 70,
  INIT_MOVER_COM_NEXT,
  INIT_NCT_CONTROLLER_OID         := 80,
  INIT_NCT_CONTROLLER_COM,
  INIT_NCT_BASE_UNIT_COUNT,
  INIT_NCT_BASE_UNIT_ITF,
  INIT_NCT_BASE_UNIT_NEXT,
  INIT_DATA_GET                   := 90,
  INIT_DATA_CHECK,
  INIT_DONE                       := 100
)UINT;
END_TYPE
```

Control / State Members

BECKHOFF

- **XtsTransport structures:**
  - E_XPU_STATE:
    - State feedback combined with E_PROGRESS

E_PROGRESS

```
{attribute 'qualified_only'}
//{attribute 'strict'}
{attribute 'to_string'}
TYPE E_PROGRESS :
(
  // progress is used in project to
  // mirror state of requested command/function
  PROGRESS_INVALID,
  PROGRESS_NOT_EXIST      :=   100,
  PROGRESS_INIT           :=  1000,
  PROGRESS_BUSY           :=  2000,
  PROGRESS_PREPARE        :=  3000,
  PROGRESS_STARTUP        :=  4000,
  PROGRESS_CHECK          :=  5000,
  PROGRESS_OCCUPIED       :=  6000,
  PROGRESS_WORKING        :=  7000,
  PROGRESS_STILL_WORKING  :=  8000,
  PROGRESS_ERROR          :=  9000,
  PROGRESS_DONE           := 10000
)UINT;
END_TYPE
```

E_XPU_STATE

```
{attribute 'to_string'}
TYPE E_XPU_STATE :
(
  XTS_NULL,
  XTS_INVALID,
  XTS_INIT                  := 10,
  XTS_STATION_INFO_READ     := 20,
  XTS_IDLE                  := 30
)UINT;
END_TYPE
```

Control / State Members

BECKHOFF

- **XtsTransport structures:**
  - E_XPU_CHECK:
    - feedback for cyclic plausibility checks to the XTS Processing Unit

E_XPU_CHECK

```
{attribute 'qualified_only'}
{attribute 'to_string'}
TYPE E_XPU_CHECK :
(
  INIT_CHECK,
  INIT_CHECK_ERROR,
  START_CHECK,
  XPU_INSTANCE_NULL,
  XPU_RAIL_LENGTH,
  GROUP_RAIL_LENGTH,
  RAIL_LENGTH_COMPARE,
  GROUP_NOT_CONFIGURED,
  GROUP_OID_MISMATCH,

  MOVER_COUNT             := 10,
  MOVER_COUNT_ZERO,
  MOVER_COUNT_NOT_EQUAL,

  PROCESSING_UNIT_POSITIONS_VALID         := 20,
  PROCESSING_UNIT_GET_DATA_EXCEED,

  MOVER_ID_STANDARD       := 30,
  MOVER_ID_SIMULATION,
  MOVER_ID_DETECTION_MODE,
  MOVER_ID_DETECTION_VALID,
  MOVER_ID_DETECTION_DC_LINK,
  MOVER_ID_DETECTION_BUSY,
  MOVER_ID_DETECTION_IDLE,
  MOVER_ID_DETECTION_STATE,
  MOVER_ID_MULTIPLE_NOT_SUPPORTED,

  POINTER_CHECK           := 90,
  POINTER_NULL_CTRL,
  POINTER_NULL_STATE
)UINT;
END_TYPE
```

Control / State Members

BECKHOFF

- **XtsTransport structures:**
  - ST_XPU_INFO:
    - Data of cyclic checks to XTS ProcessingUnit

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;

  nParameterSetCount   : UINT;

END_STRUCT
END_TYPE
```

Control / State Members

BECKHOFF

- **XtsTransport structures:**
  - ST_GROUP_INFO:
    - Data of cyclic checks to Collision Avoidance Group (CAGroup)

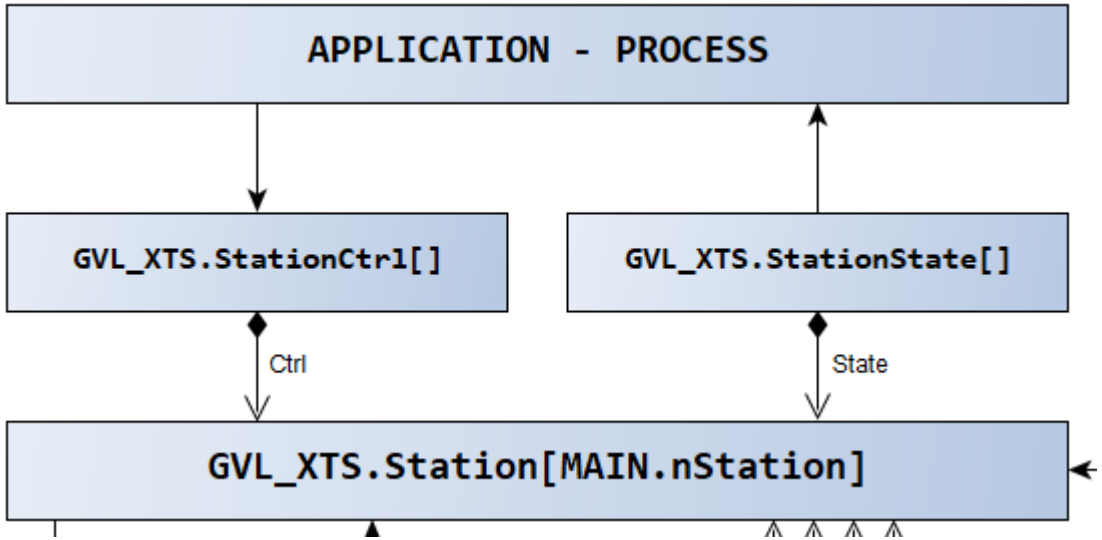
ST_GROUP_INFO

```
1  TYPE ST_GROUP_INFO :
2  STRUCT
3    GroupStatusValid,
4    GroupStatusBusy,
5    GroupMoving,
6    GroupHoming,
7    GroupErrorStop,
8    GroupNotReady,
9    GroupStandby,
10   GroupStopping,
11
12   GroupDisabled,
13   AllAxesStanding,
14   ConstantVelocity,
15   Accelerating,
16   Decelerating,
17   InPosition,
18   GroupError            : BIT;
19
20   GroupErrorId          : UDINT;
21
22   AxisCount             : UDINT;
23   AxisCountEnabled      : UDINT;
24
25   {attribute 'displaymode':='hex'}
26   CaGroupOID            : OTCID;
27
28   CaGroupState          : E_CA_GROUP_STATE;
29 END_STRUCT
30 END_TYPE
31
```

Control / State Members

BECKHOFF

- **XtsStation structures:**
  - ST_STATION_CTRL:
    - Command for handshaking mover transport
    - Command parameter for sending mover to a target station.
  - ST_STATION_STATE:
    - State for handshaking mover transport
    - State details about current mover



Control / State Members

BECKHOFF

- **XtsStation structures:**
  - ST_STATION_CTRL:
    - E_STATION_CTRL:
      - Command for handshaking mover transport in station
    - Command parameter for sending mover to a target station.
      - nMask sets index of PosStop[] in target
      - nTargetStation sets index of target station
      - rOffset sets optional mover offset in target station

ST_STATION_CTRL

```
1  {attribute 'pack_mode' := '2'}
2  TYPE ST_STATION_CTRL :
3  STRUCT
4    eCmd            : E_STATION_CTRL;
5    {attribute 'displaymode':='bin'}
6    nMask           : BYTE;
7    nTargetStation  : USINT;
8    rOffset         : REAL;
9
10 END_STRUCT
11 END_TYPE
12
```

Control / State Members

BECKHOFF

- **XtsStation structures:**
  - ST_STATION_STATE:
    - E_STATION_STATE:
      - State for handshaking mover transport in station
    - nMask:
      - In case of ConfiguredStopCount > 1 nMask shows current index of PosStop[] in station
    - nMoverId:
      - Active mover in station
    - rMoverModPos:
      - Modulo position of mover in station
    - nQueue
      - Count of movers which were sent to station

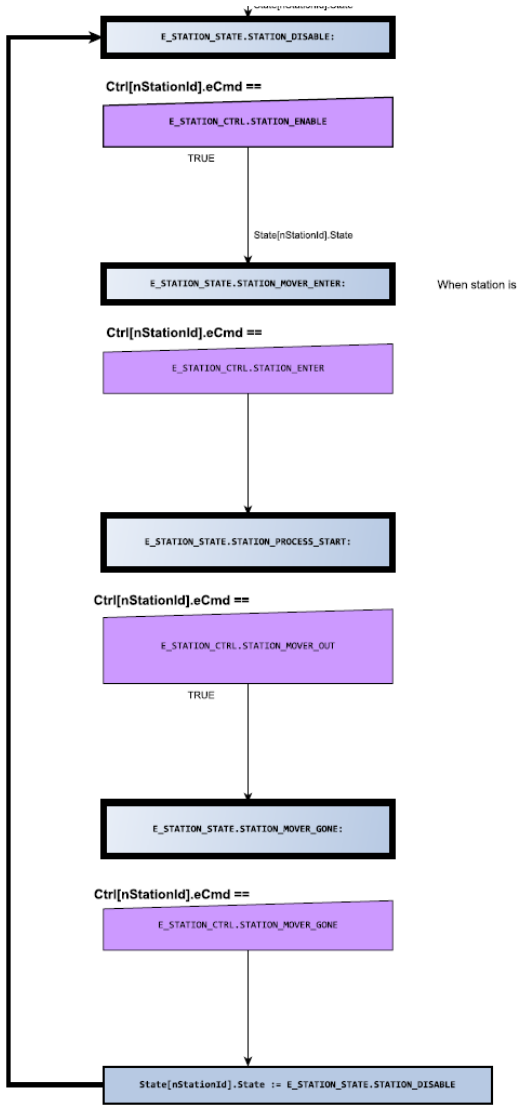
ST_STATION_STATE

```
1  {attribute 'pack_mode' := '2'}
2  TYPE ST_STATION_STATE :
3  STRUCT
4    eState          : E_STATION_STATE;
5    {attribute 'displaymode':='bin'}
6    nMask           : BYTE;
7    nMoverId        : USINT;
8    rMoverModPos    : LREAL;
9    nQueue          : USINT;
10 END_STRUCT
11 END_TYPE
12
```

Control / State Members

BECKHOFF

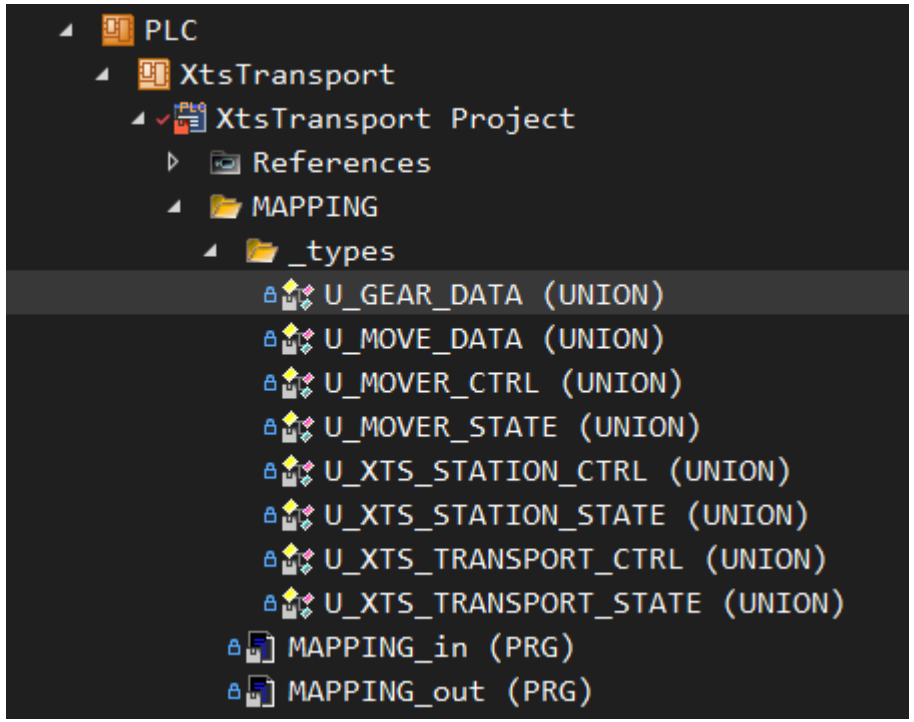
- **XtsStation structures:**
  - E_STATION_CTRL / E_STATION_STATE
    - Handshake procedures you'll find in example pdfs in the doc folder of this project.



Implementation

BECKHOFF

- **Mapping structures:**
  - **YOU can decide which structures you need for your application**
  - Use of UNIONS in TwinCAT
  - Dedicated PRGs for mapping
    - Mapping_In
    - Mapping_Out
  - All interfacing structures → pack mode := 2
  - All fieldbus mappings as array of byte



Implementation

BECKHOFF

- **Profinet:**
  - **EL6631-0010 Profinet Decive terminal**
    - **See Beckhoff Infosys for documentation.**
    - https://infosys.beckhoff.com/index.php

Technical data	Supplement
PROFINET Version	RT Class 1 ConformanceClassB
Number of device interfaces)[1]	8
Topology	variable
Quantity of user data	per device, maximum one Ethernet frame length)[2] 1500 bytes of user data, inc. IOPS and IOCS
Cycle time	≥ 1 ms

)[1] see the chapter on virtual PROFINET devices
)[2] Depending on the cycle time, the PROFINET cycle time, and the CPU being used

Conditions required for operation

The following points must be observed when using the PROFINET supplement:

- Only Ethernet cards with Intel chipset allowed.
- RealTime Ethernet driver must be installed.
- No other RealTime protocols must be connected through this interface.
- The real-time capability can only be guaranteed in the transmit direction; in the receive direction, the possibility of incorrect use means that it cannot be guaranteed. This might, for example, be the copying of large quantities of data through this interface.
  It is recommended that the PROFINET network is separated from other networks.

- **EAP:**
  - The EtherCAT Automation Protocol (EAP) device enables the cyclic, highly deterministic exchange of any desired variables between PCs that are connected by Ethernet. Communication between EAP devices takes place according to the Publisher/Subscriber principle and is specified by the EtherCAT Technology Group (ETG) (ETG 1005 – see webpage www.ethercat.org).
  - The real-time Ethernet driver for TwinCAT must be installed for the TwinCAT EAP device in order for highly deterministic communication to take place
  - https://infosys.beckhoff.com/index.php

Implementation

BECKHOFF

- **Mapping structures:**

U_XTS_TRANSPORT_CTRL | U_XTS_TRANSPORT_STATE

```
1  {attribute 'pack_mode' := '2'}
2  TYPE U_XTS_TRANSPORT_CTRL :
3  UNION
4    stCtrl              : ST_XTS_TRANSPORT_CTRL;
5    byCtrl    AT %I*    : ARRAY[1..SIZEOF(ST_XTS_TRANSPORT_CTRL)] OF BYTE;
6  END_UNION
7  END_TYPE
```

U_XTS_TRANSPORT_CTRL | U_XTS_TRANSPORT_STATE

```
1  {attribute 'pack_mode' := '2'}
2  TYPE U_XTS_TRANSPORT_STATE :
3  UNION
4    stState             : ST_XTS_TRANSPORT_STATE;
5    byState   AT %I*    : ARRAY[1..SIZEOF(ST_XTS_TRANSPORT_STATE)] OF BYTE;
6  END_UNION
7  END_TYPE
```

- **Mapping structures:**

U_XTS_STATION_CTRL | U_XTS_STATION_STATE

```
1  TYPE U_XTS_STATION_CTRL :
2  UNION
3    stCtrl              : ARRAY[1..MAX_STATION] OF ST_STATION_CTRL;
4
5    byCtrl    AT %I*    : ARRAY[1..MAX_STATION] OF
6                          ARRAY[1..SIZEOF(ST_STATION_CTRL)] OF BYTE;
7  END_UNION
8  END_TYPE
9
```

U_XTS_STATION_CTRL | U_XTS_STATION_STATE

```
1  TYPE U_XTS_STATION_STATE :
2  UNION
3    stState             : ARRAY[1..MAX_STATION] OF ST_STATION_STATE;
4
5    byState   AT %I*    : ARRAY[1..MAX_STATION] OF
6                          ARRAY[1..SIZEOF(ST_STATION_STATE)] OF BYTE;
7  END_UNION
8  END_TYPE
9
```

Implementation

BECKHOFF

- **Mapping structures:**

U_MOVER_CTRL | U_MOVER_STATE | U_MOVE_DATA | U_GEAR_DATA

```
1  TYPE U_MOVER_CTRL :
2  UNION
3    stCtrl              : ARRAY[1..MAX_MOVER] OF ST_MOVER_CTRL;
4
5    byCtrl    AT %I*    : ARRAY[1..MAX_MOVER] OF
6                          ARRAY[1..SIZEOF(ST_MOVER_CTRL)] OF BYTE;
7  END_UNION
8  END_TYPE
9
```

U_MOVER_CTRL | U_MOVER_STATE | U_MOVE_DATA | U_GEAR_DATA

```
1  TYPE U_MOVER_STATE :
2  UNION
3    stState             : ARRAY[1..MAX_MOVER] OF ST_MOVER_STATE;
4
5    byState   AT %Q*    : ARRAY[1..MAX_MOVER] OF
6                          ARRAY[1..SIZEOF(ST_MOVER_STATE)] OF BYTE;
7  END_UNION
8  END_TYPE
9
```

Implementation

BECKHOFF

- **Mapping structures:**

U_MOVER_CTRL | U_MOVER_STATE | **U_MOVE_DATA** | U_GEAR_DATA

```
1  TYPE U_MOVE_DATA :
2  UNION
3    stData              : ARRAY[1..MAX_MOVER] OF ST_MOVE_DATA;
4
5    byData    AT %I*    : ARRAY[1..MAX_MOVER] OF
6                          ARRAY[1..SIZEOF(ST_MOVE_DATA)] OF BYTE;
7  END_UNION
8  END_TYPE
9
```

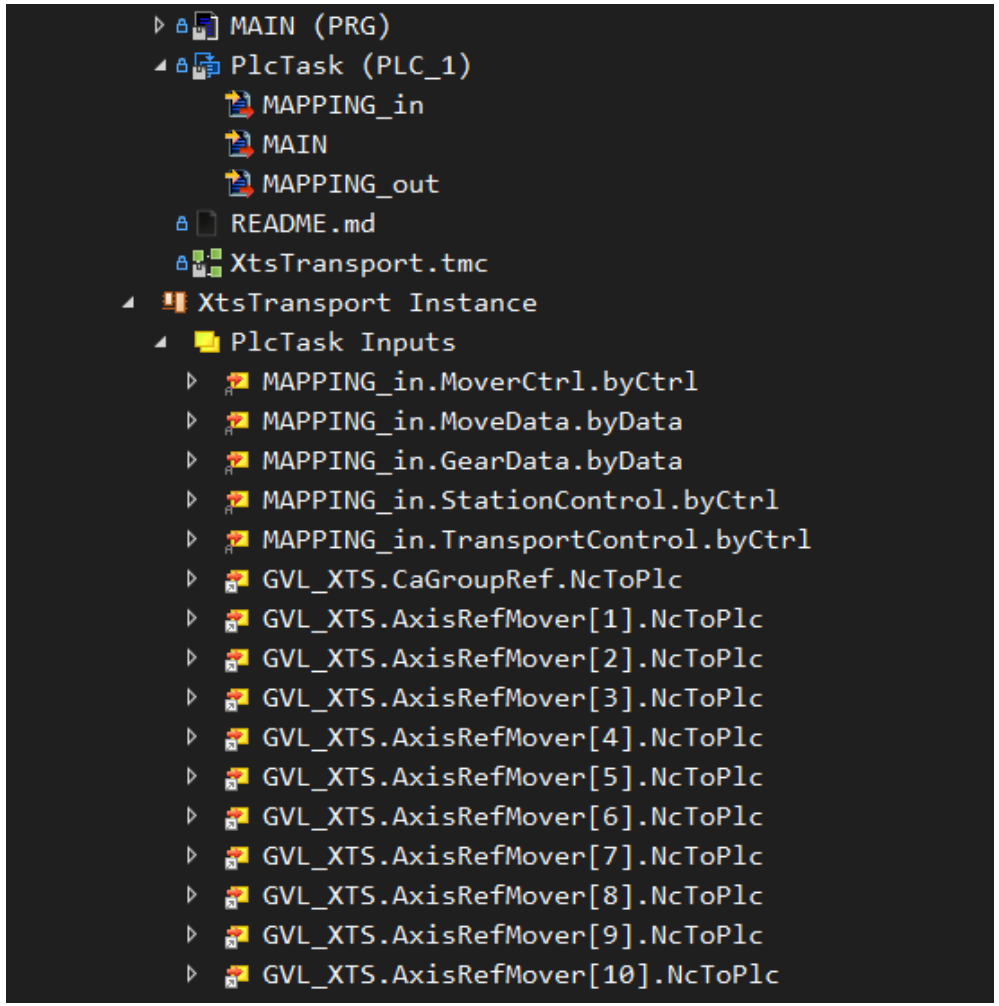
U_MOVER_CTRL | U_MOVER_STATE | U_MOVE_DATA | **U_GEAR_DATA**

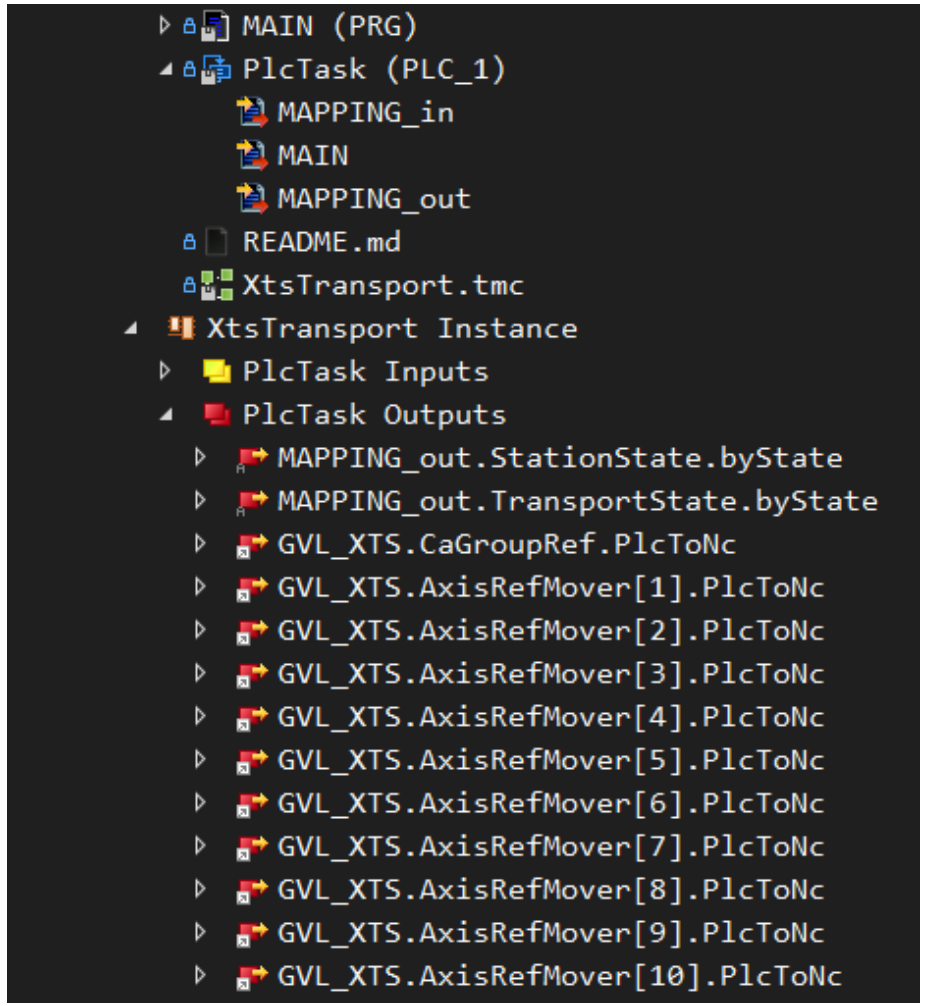
```
1  TYPE U_GEAR_DATA :
2  UNION
3    stData              : ARRAY[1..MAX_MOVER] OF ST_GEAR_DATA;
4
5    byData    AT %I*    : ARRAY[1..MAX_MOVER] OF
6                          ARRAY[1..SIZEOF(ST_GEAR_DATA)] OF BYTE;
7  END_UNION
8  END_TYPE
9
```

Implementation

BECKHOFF

- **Mapping example:**
  - **5 Stations**
  - **10 Mover**

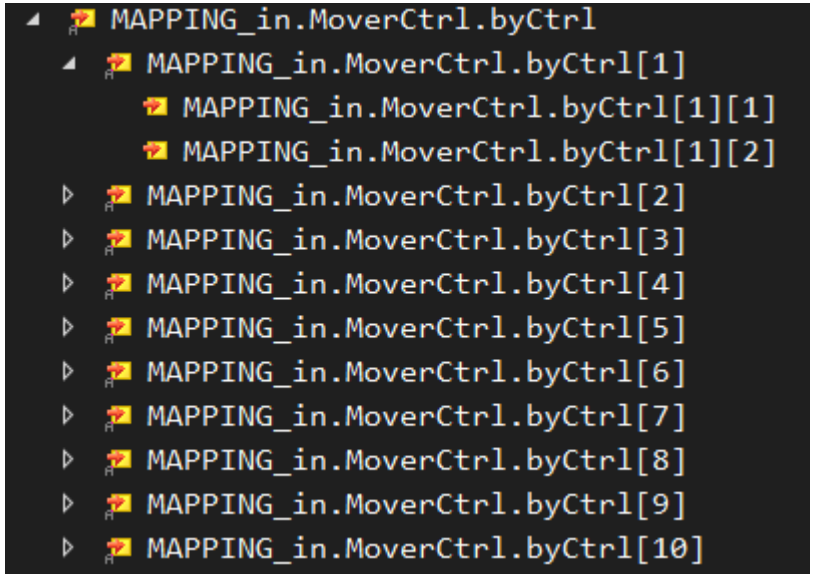


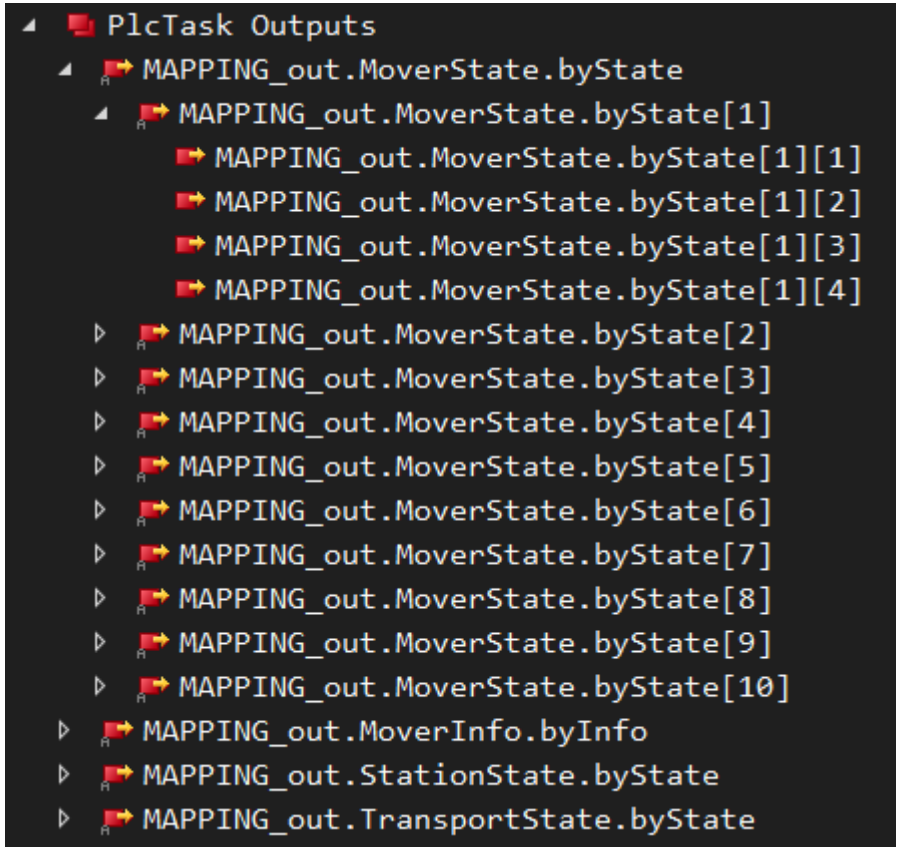


Implementation

BECKHOFF

- **Mapping example:**
  - **5 Stations**
  - **10 Mover**





Implementation

BECKHOFF

- **Mapping example:**
  - **5 Stations**
  - **10 Mover**