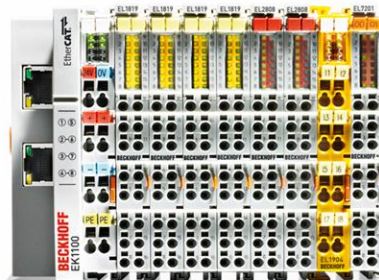


PLC_MOTION_LAYER

TwinCAT project

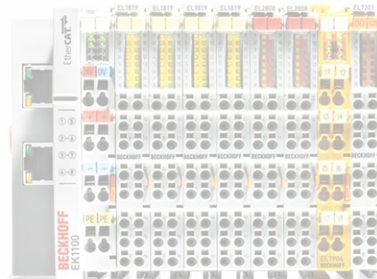
BECKHOFF



Daniel Hauer

Introduction to motion layer approach

- motivation for a motion layer
- use cases
- specific construction



Motivation:

- **Centralized control:**
 - Sync communication
 - Async communication
 - Agnostic to communication technology

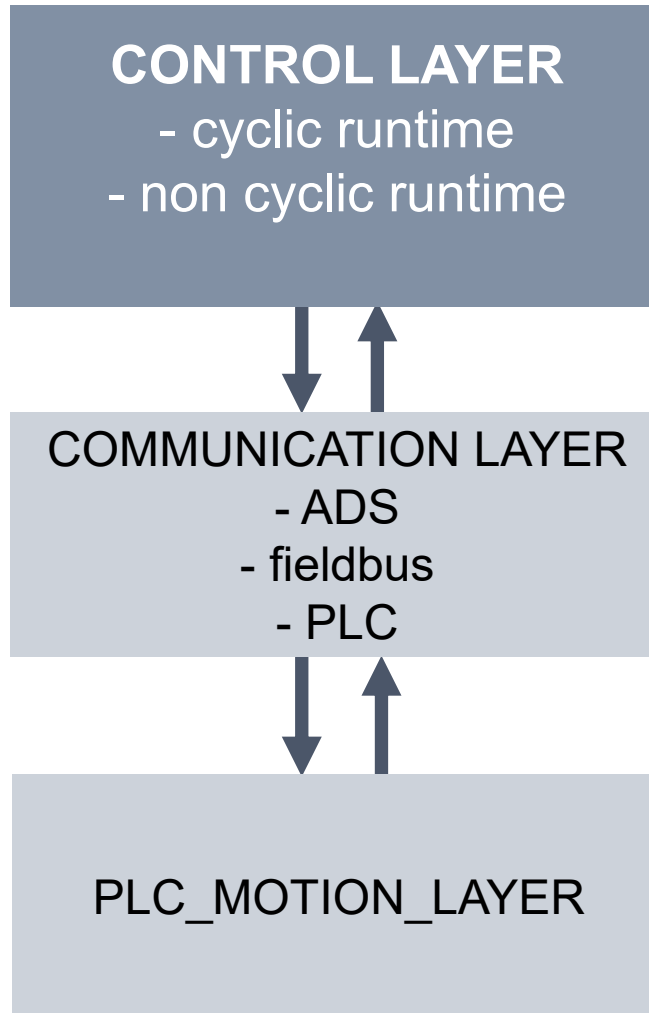
- **Decentralized execution of motion tasks**
 - Data consistency in cyclic multi task environments
 - Local machine with modular design specifications
 - Unified procedure for modular software architecture
 - Use of top layer consistent through different architectures

Motivation:

- **Use of TwinCAT supported/updated libraries**
 - Tc2_MC2
 - Tc2_MC2_Drive
 - Tc2_NC
 - Tc2_NCI
 - Tc2_PlcInterpolation
- **Open code base**
 - Migration to Tc3 MC in preparation
 - Customer/user specific changes possible
 - Access to code
 - Conversion to library possible by customer/user
- **Compiled PLC**
 - Source code is not on shipped machine

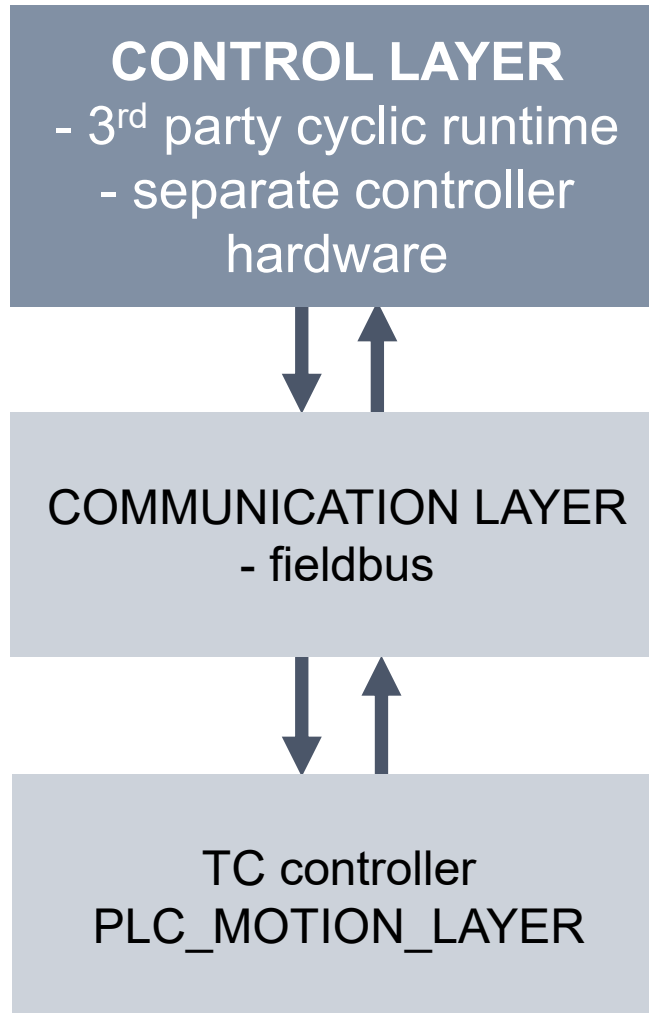
Motivation:

- Use of TwinCAT motion without detailed coding knowledge
- Transparency of communication layer
- Code base shall remain independent of control layer
- Configurable Options for specific libraries / TC functions
- Balanced load for configurable options in machine layout
- Stable cpu use for XFC applications



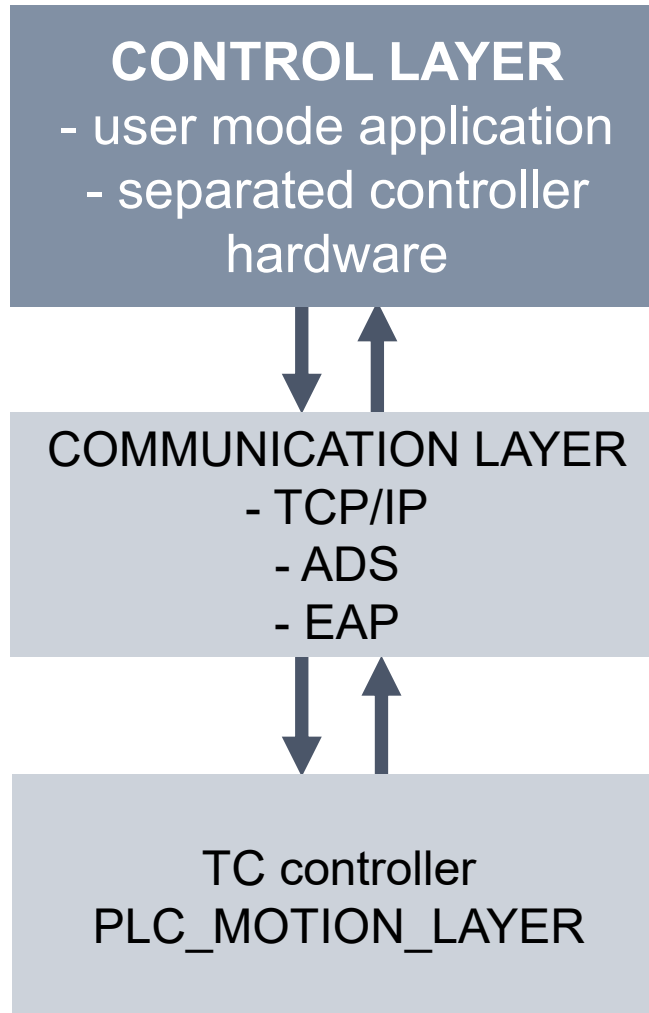
Use cases:

- Separate controller for machine logic
- Any fieldbus (EtherCAT, Profi...)
- Connected through TwinCAT mappings
- Execution of motion tasks in PLC_MOTION_LAYER TwinCAT controller



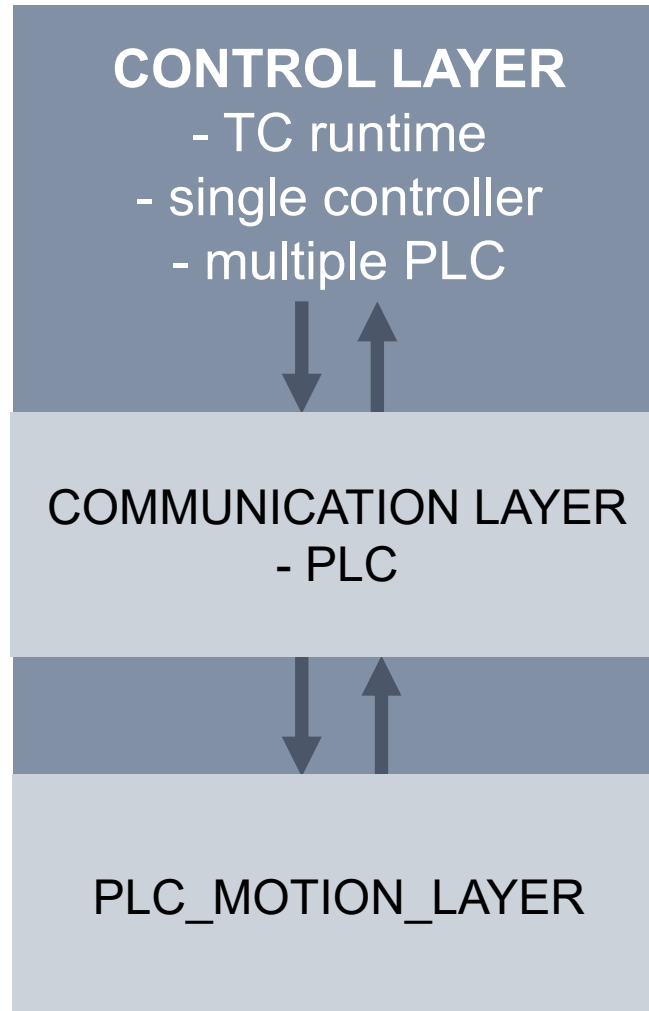
Use cases:

- Separate controller for machine logic
- Any network
- Connected through TwinCAT mappings
- Execution of motion tasks in PLC_MOTION_LAYER TwinCAT controller



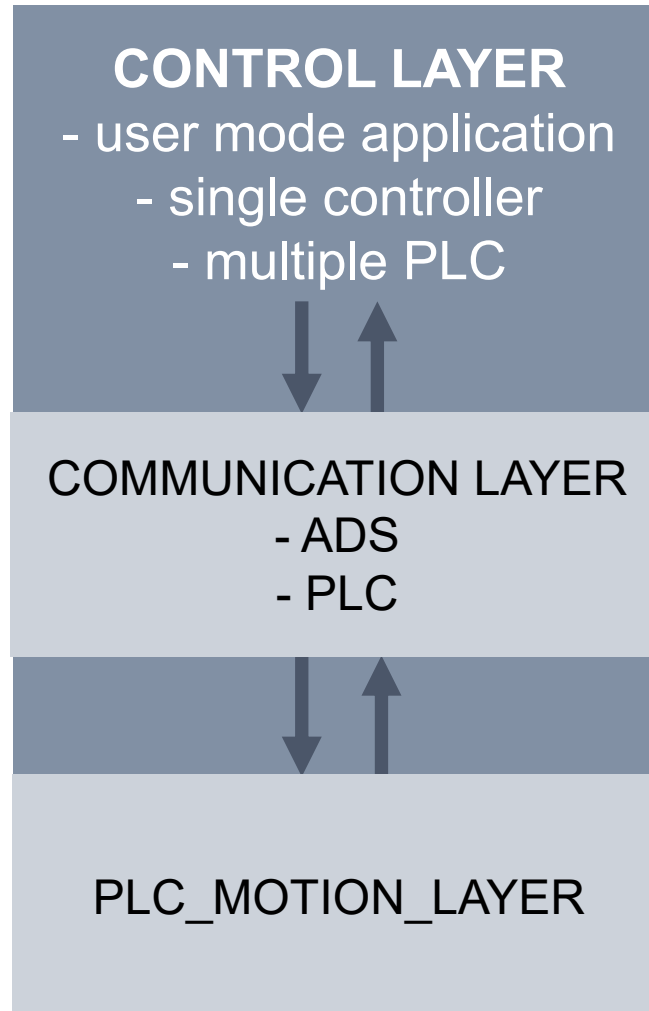
Use cases:

- One controller for machine logic
- Multiple PLC for machine logic
- Connected through TwinCAT mappings
- Execution of motion tasks



Use cases:

- One controller for machine logic
- User mode application AND/OR multiple PLC
- ADS for symbol access by user mode application
- TwinCAT mapping for connecting multiple PLCs for specific application purposes
- Execution of motion tasks



Specific construction:

- TwinCAT PLC project
- Use of specific syntactic code behaviour
- Software design
- Compiler defines / pragmas
- Logging system

TwinCAT project:

- Default TwinCAT project
 - Adjust core settings to target hardware
 - Add NC/PtP
 - Optional add NCI channel
- TwinCAT PLC
 - Add existing Item: PLC_MOTION_LAYER
 - Add task reference
 - Option: add compiler defines (NCI, CAM, BSD)
- Adjust constants in:
 - PLC_MOTION_LAYER/PLC_CONSTANT
- Compile
 - PLC_MOTION_LAYER Instance mapping is built

specific syntactic code behaviour:

- C like state machines
 - State changes need not consume one PLC cycle
 - Same cycle response to command on cyclic interface
- OnChange detection for new commands
 - Cyclic check whether the command has changed
- State always carries offset about progress of command (busy, error, done)
- Instance FBs are called within states

Software Design:

- Every TwinCAT function has dedicated wrapper
 - Separate namespaces
 - Optional library binding
- Cross communication via interfaces
 - NCI, CAM, XFC use interfaces in order to enable optional binding
 - If compiler define is not set, empty interfaces are used instead of instances
- Ctrl/State structures for commanding required function
 - PtP – ctrl/state
 - NCI – ctrl/state
 - CAMMING – ctrl/state
- Parameter structures carry required data for commanded function
 - PtP – (SetPos, SetVelo, SetAcc, MasterAxisIndex...)
 - NCI – (AxisGroupId, AxisIndex, MFunc, RParameter...)
 - CAMMING – (MasterAxisIndex, TableId...)

Software Design:

PLC_MOTION_LAYER / GVL_AXIS

```
//-----  
// command and state structure  
//-----  
Ctrl  : ARRAY[1..MAX_AXIS] OF  
        ST_AXIS_CTRL;  
  
State : ARRAY[1..MAX_AXIS] OF  
        ST_AXIS_STATE;  
  
//-----  
// cyclic interface function block  
//-----  
Control : ARRAY[1..MAX_AXIS] OF  
        FB_McAxisCtrl;
```

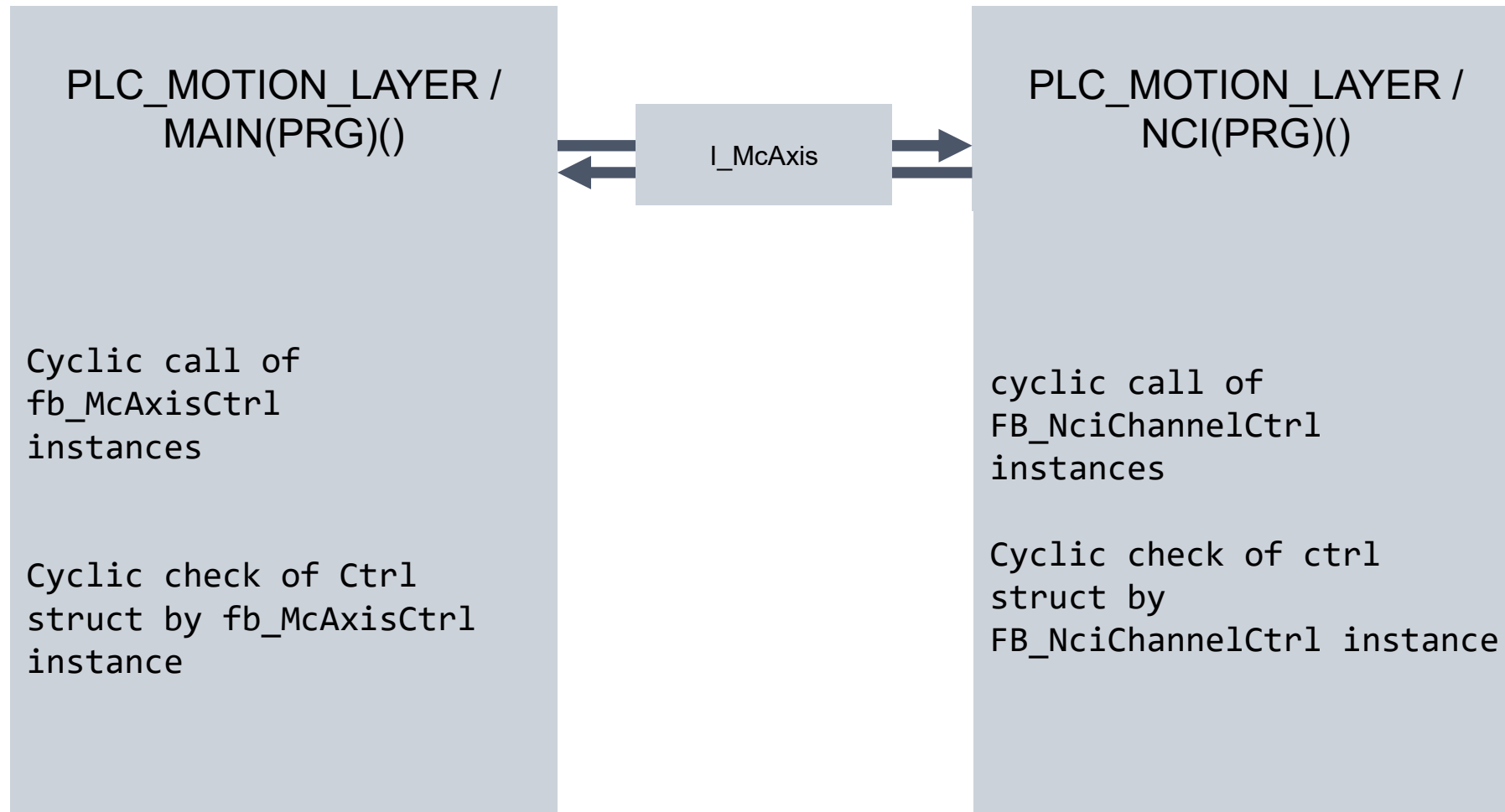
PLC_MOTION_LAYER / GVL_NCI

```
//-----  
// command and state structure  
//-----  
stChannelCtrl : ARRAY[1..MAX_NCI_CH] OF  
        ST_CTRL_NCI;  
  
stChannelState : ARRAY[1..MAX_NCI_CH] OF  
        ST_STATE_NCI;  
  
//-----  
// cyclic interface function block  
//-----  
fbNciCtrl : ARRAY[1..MAX_NCI_CH] OF  
        FB_NciChannelCtrl;
```

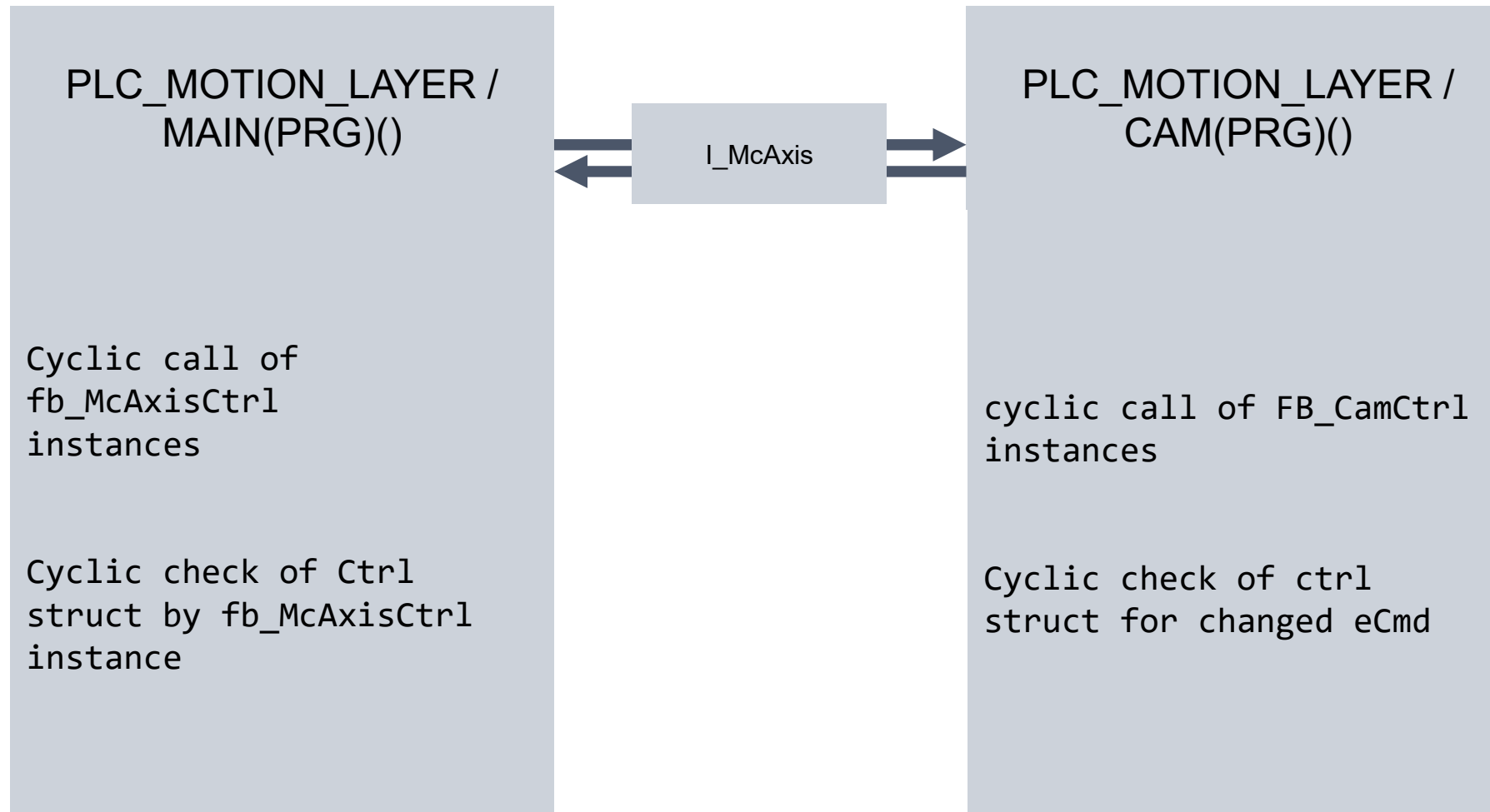
PLC_MOTION_LAYER / GVL_CAM

```
//-----  
// command and state structure  
//-----  
Ctrl  : ARRAY[1..MAX_AXIS] OF  
        ST_CAM_CTRL;  
  
State : ARRAY[1..MAX_AXIS] OF  
        ST_CAM_STATE;  
  
//-----  
// cyclic interface function block  
//-----  
Control : ARRAY[1..MAX_AXIS] OF  
        FB_CamCtrl;
```

Software Design:

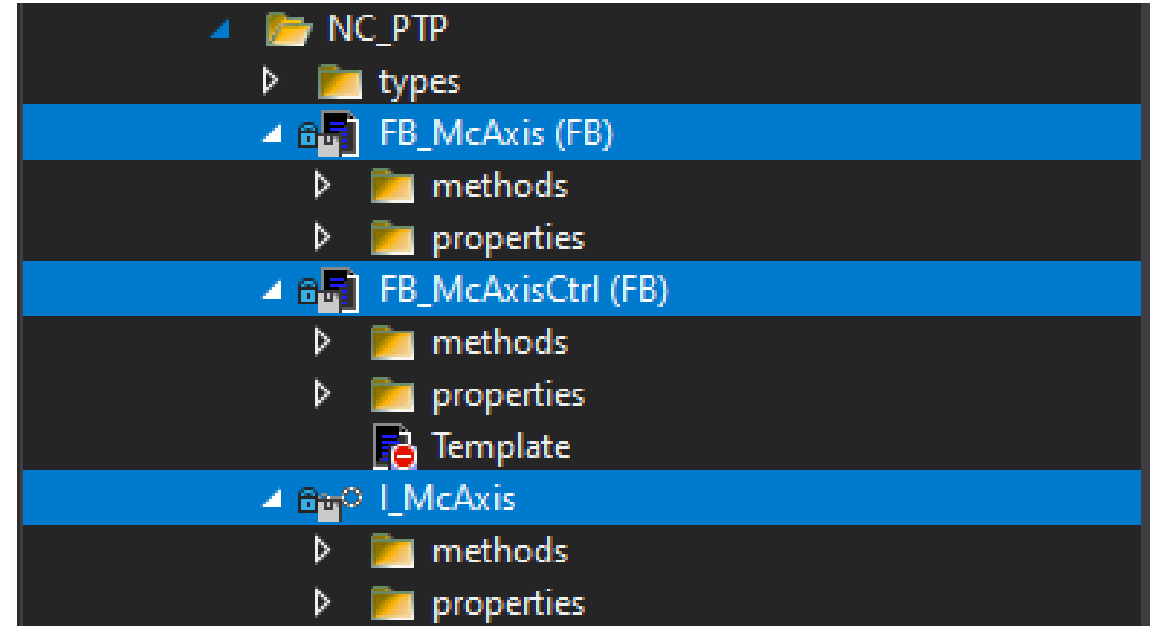


Software Design:



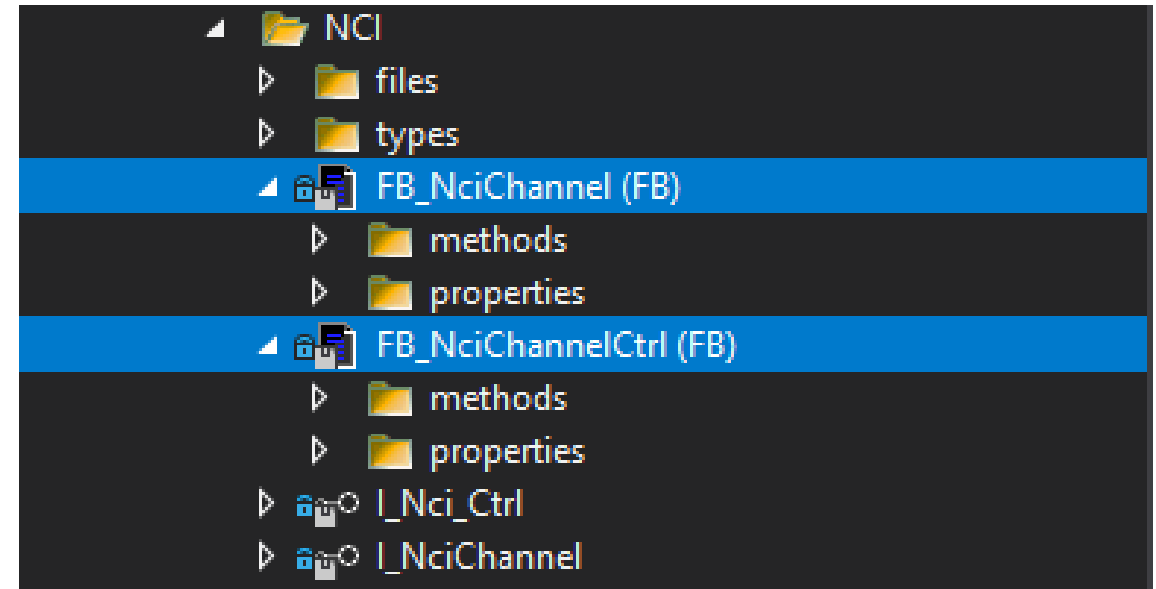
GVL_AXIS:

- McAxis: Point To Point axis
 - Base class **FB_McAxis** wraps Tc2_MC2 function blocks and implements interface
 - **FB_McAxisCtrl** extends base class with cyclic execution wrapper
 - **I_McAxis** is used in advanced motion features (NCI, CAM, XFC)



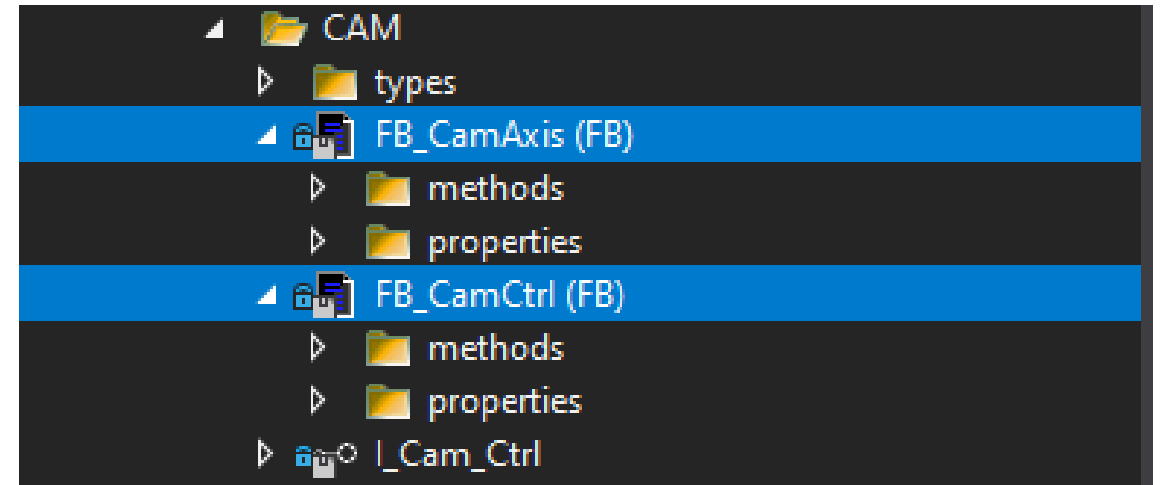
GVL_NCI:

- NCI Channel: XYZ interpolated
 - Base class **FB_NciChannel** wraps Tc2_NCI function blocks and implements interface
- **FB_NciChannelCtrl** extends base class with cyclic execution wrapper and implements interface
- **I_NciChannel** and **I_Nci_Ctrl** are only valid if compiler define is set before compiling project



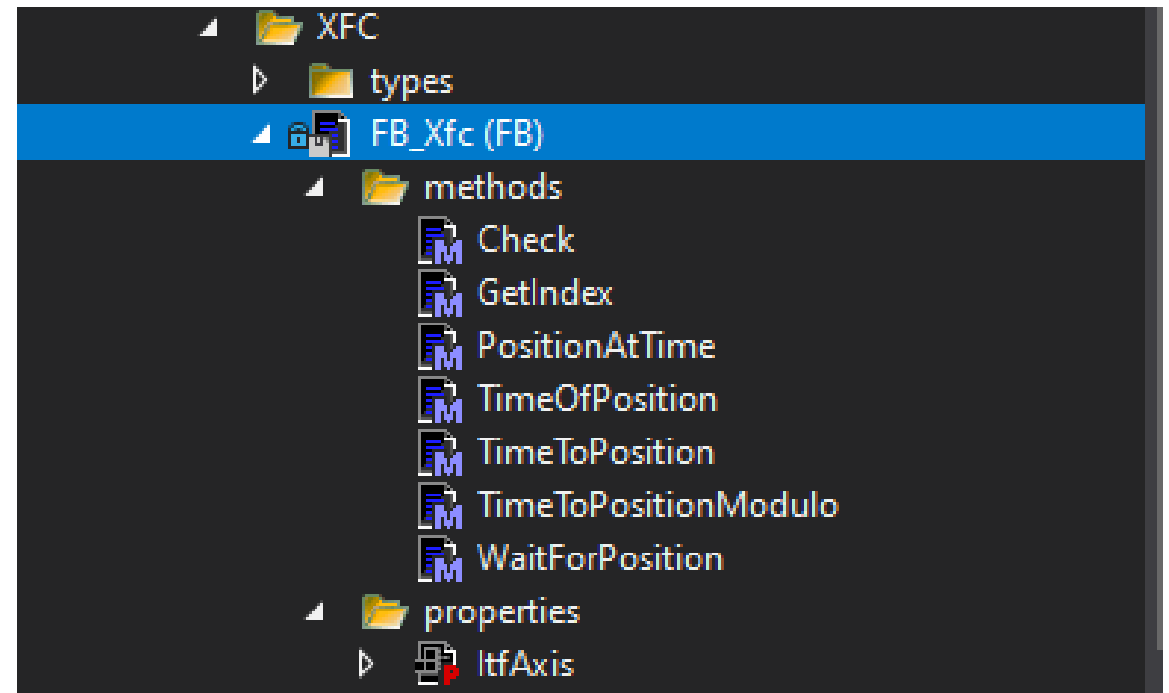
GVL_CAM:

- Camming:
 - Base class **FB_CamAxis** wraps Tc2_MC2_Camming function blocks and implements interface
 - **FB_CamCtrl** extends base class with cyclic execution wrapper and implements interface
 - **I_Cam_Ctrl** is only valid if compiler define is set before compiling project



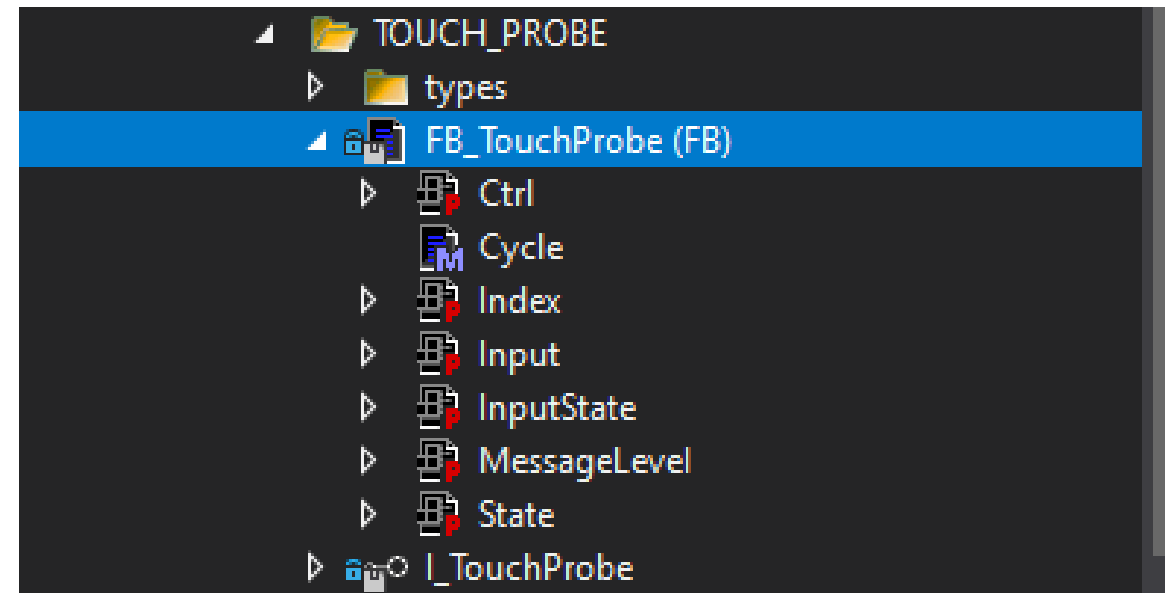
GVL_FUNCTIONS:

- XFC classes:
 - Base class **FB_Xfc** wraps Tc2_MC2_XFC function blocks for Distributed Clock position/time applications



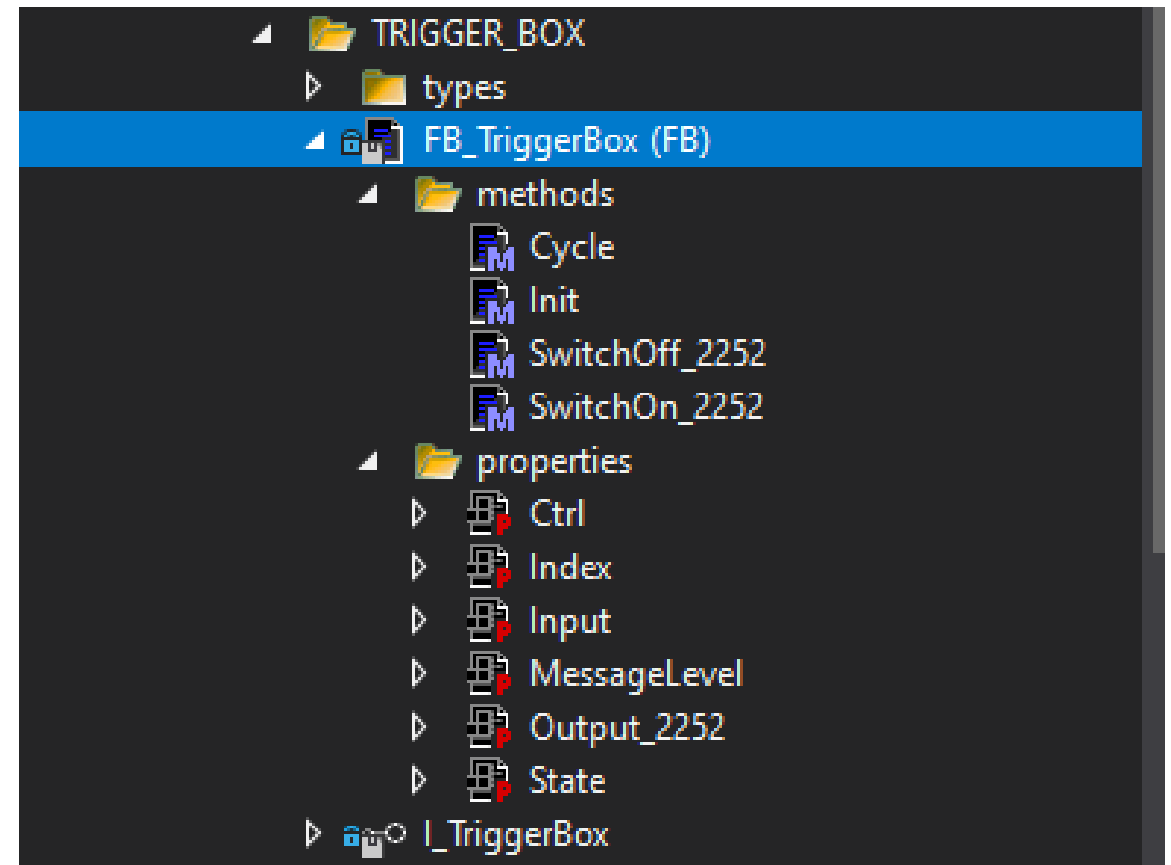
GVL_FUNCTIONS:

- Touch Probe:
 - **FB_TouchProbe** extends **FB_Xfc**
 - Cyclic execution with Ctrl/State pair
 - Must be connected to input device
 - **I_TouchProbe** is only valid if compiler define is set before compiling project



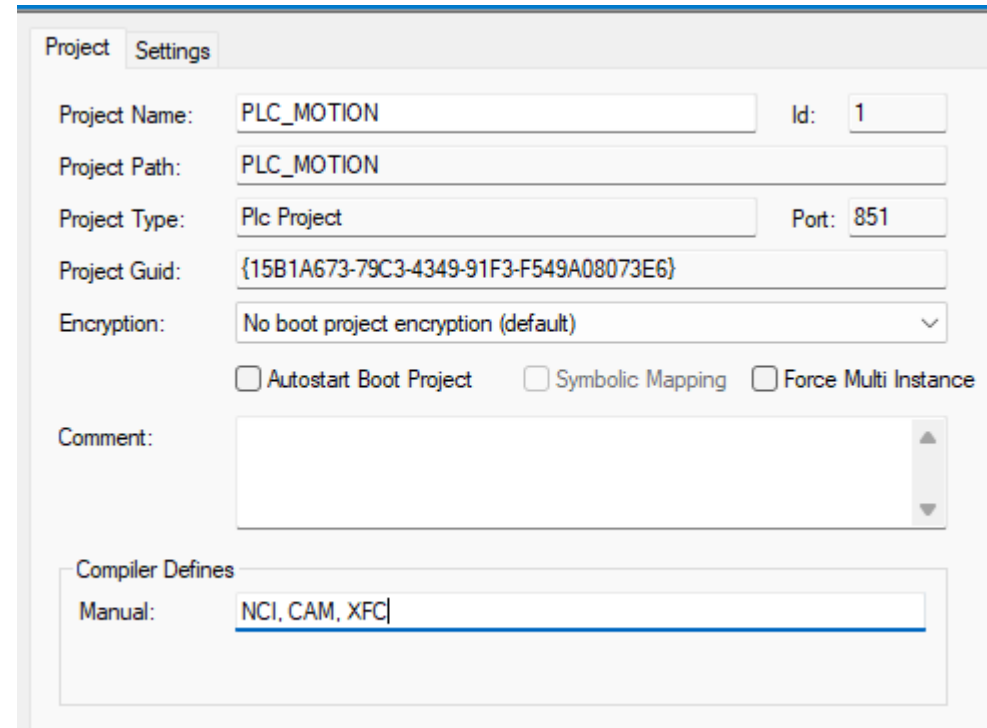
GVL_FUNCTIONS:

- Trigger Box:
 - **FB_TriggerBox** extends **FB_Xfc**
 - Cyclic execution with Ctrl/State pair
 - Must be connected to input/output device
 - **I_TriggerBox** is only valid if compiler define is set before compiling project



Compiler defines / pragmas:

- **BSD**
 - TC-BSD system specific variables are set
- **NCI**
 - NciChannel cyclic interfaces are used
- **CAM**
 - Camming cyclic interfaces are used
- **XFC**
 - TouchProbe and TriggerBox interfaces are used



The screenshot shows the 'Settings' tab of a project configuration window. The 'Project Name' is 'PLC_MOTION' and the 'Id' is '1'. The 'Project Path' is also 'PLC_MOTION'. The 'Project Type' is 'Plc Project' and the 'Port' is '851'. The 'Project Guid' is '{15B1A673-79C3-4349-91F3-F549A08073E6}'. The 'Encryption' is set to 'No boot project encryption (default)'. There are three checkboxes: 'Autostart Boot Project', 'Symbolic Mapping', and 'Force Multi Instance', all of which are currently unchecked. There is a 'Comment' text area which is empty. At the bottom, under 'Compiler Defines', the 'Manual' field contains the text 'NCI, CAM, XFC|'.

Project Name:	PLC_MOTION	Id:	1
Project Path:	PLC_MOTION		
Project Type:	Plc Project	Port:	851
Project Guid:	{15B1A673-79C3-4349-91F3-F549A08073E6}		
Encryption:	No boot project encryption (default)		
<input type="checkbox"/> Autostart Boot Project <input type="checkbox"/> Symbolic Mapping <input type="checkbox"/> Force Multi Instance			
Comment:			
Compiler Defines			
Manual:	NCI, CAM, XFC		

Compiler defines / pragmas:

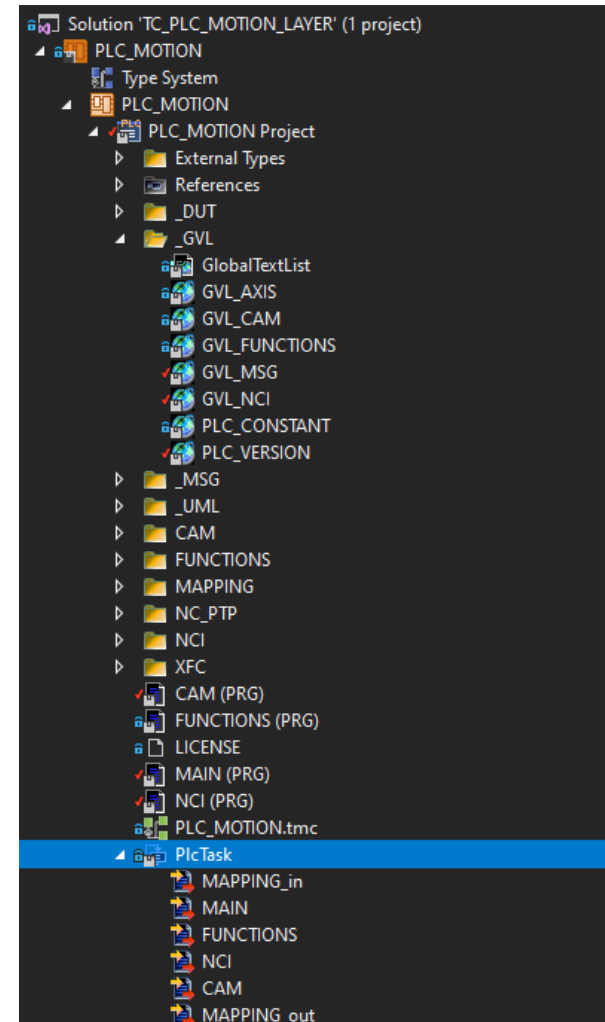
- **AXIS_MAP**
 - MAPPING_in.AxisCtrl is copied onto GVL_AXIS.Ctrl and data structures
 - GVL_AXIS.State is copied onto MAPPING_out.AxisState and info structures

- **CAM_MAP**
 - MAPPING_in.CamControl is copied onto GVL_CAM.Ctrl
 - GVL_CAM.State is copied onto MAPPING_out.CamState and info structures

- **TRIGGER_MAP**
 - TouchProbe and TriggerBox interfaces are used

Cyclic call tree:

- **MAPPING_in(PRG)**
 - Copies mapping data onto input structures
- **MAIN(PRG)**
 - Cyclic call to FB_McAxisCtrl instances
- **FUNCTIONS(PRG)**
 - Cyclic call to TouchProbe and TriggerBox instances
- **NCI(PRG)**
 - Cyclic call to NCI channel instances
- **CAM(PRG)**
 - Cyclic call to FB_CamCtrl instances
- **MAPPING_out**
 - Copies state data onto mapping structures



Logging System:

- Implementation from top down
 - → function based with global timestamp added automatically
- Enumeration based with 4 categories and timestamp
 - → occurrence in strict timestamp order
- Error Id is mirrored directly from called instance
 - → Infosys error numbers can be searched in case of diagnosis
- Optional text for additional information
- Specific logging switch to get more detailed information aside error numbers
- Automated write procedure to ascii formatted file

PLC_MOTION_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.