SC52

APPLICATION AND LIBRARIES

EPEC

SIL2 APPLICATION FEATURES

- SC52 application includes both safety related and non-safety related parts
- The code template includes two tasks
  - Safety related task (*SafePRG_Task*)
    - Calls safety application code (*S_PLC_PRG*)
  - Non-safety related task (*NonSafePRG_Task*)
    - Calls non-safety related application code (*PLC_PRG*)

EPEC

TASK CONFIGURATION

- Safety related task *SafePRG_Task*
  - Calls *S_PLC_PRG*, all safety related application code is run in this task
  - Priority 0
  - Cycle time 10 ms
  - Watchdog 10 ms

SafePRG_TASK has priority over *NonSafePRG_TASK*.

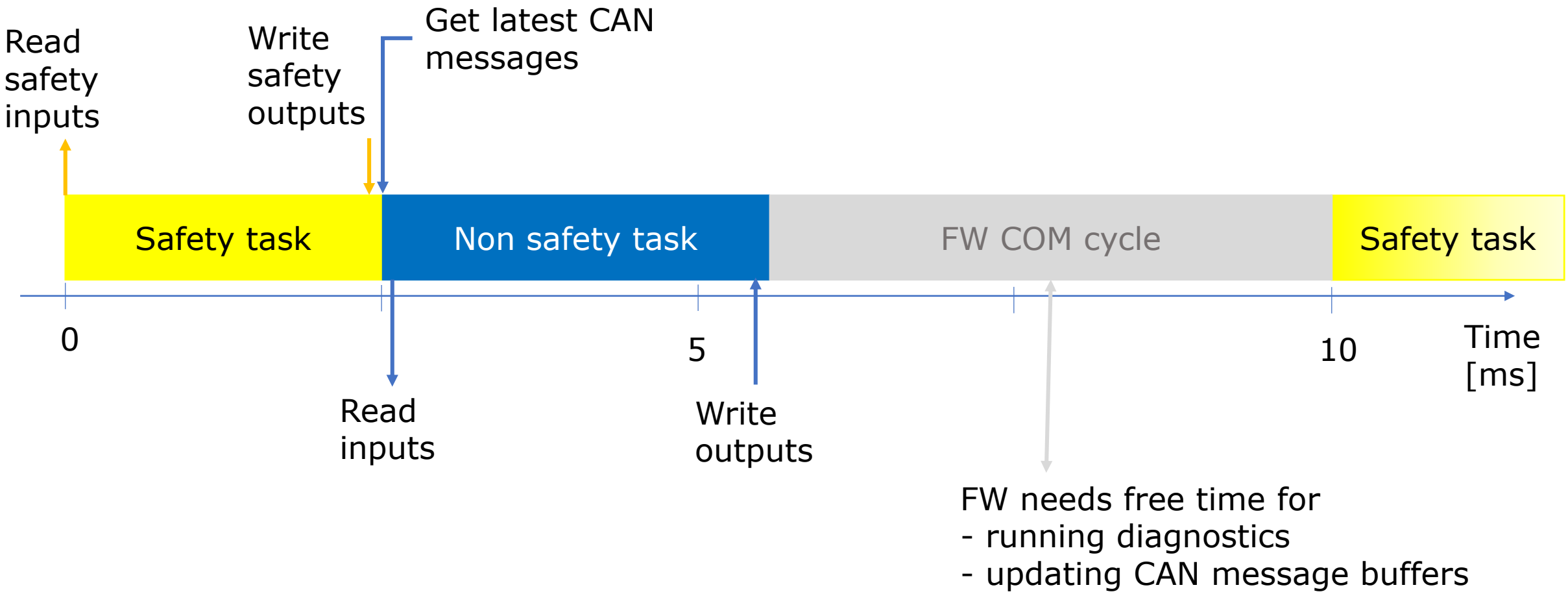
- Non-safety related task *NonSafePRG_Task*
  - Calls *PLC_PRG*, all non-safety related application code is run in this task
  - Priority 1
  - Cycle time 10 ms
  - Watchdog not set by default

TASK CONFIGURATION

- Safety related task will interrupt lower priority task execution
  - *SafePRG_TASK* priority value is lower (better priority) than *NonSafePRG_TASK*
- Only safety related task cycle time can be guaranteed to be steady (higher priority)

- Note!
  - Safety and non-safety related task total cycle time should be less than defined time in *Task Configuration* (default 10ms)
  - Firmware runs its own COM cycle task between application tasks

Task configuration



SAFETY RELATED DATA IN NON-SAFETY RELATED MEMORY?

- Two different ways
  1. Duplicated (diversity) data
     - I/O: using two different input pins or one Cat 2 input
     - SRDO: two messages, plain and inverted data
  2. Checksum calculation
     - safety related parameters

- **The data from non-safe memory should always be validated before using it in safety application**

Safety Related POU

- Variables, including I/O pins, can be safety related or non-safety related
- SAFE prefix in variable type is only for coloring purpose.
  - Does not affect the variable context (safety/non-safety related)

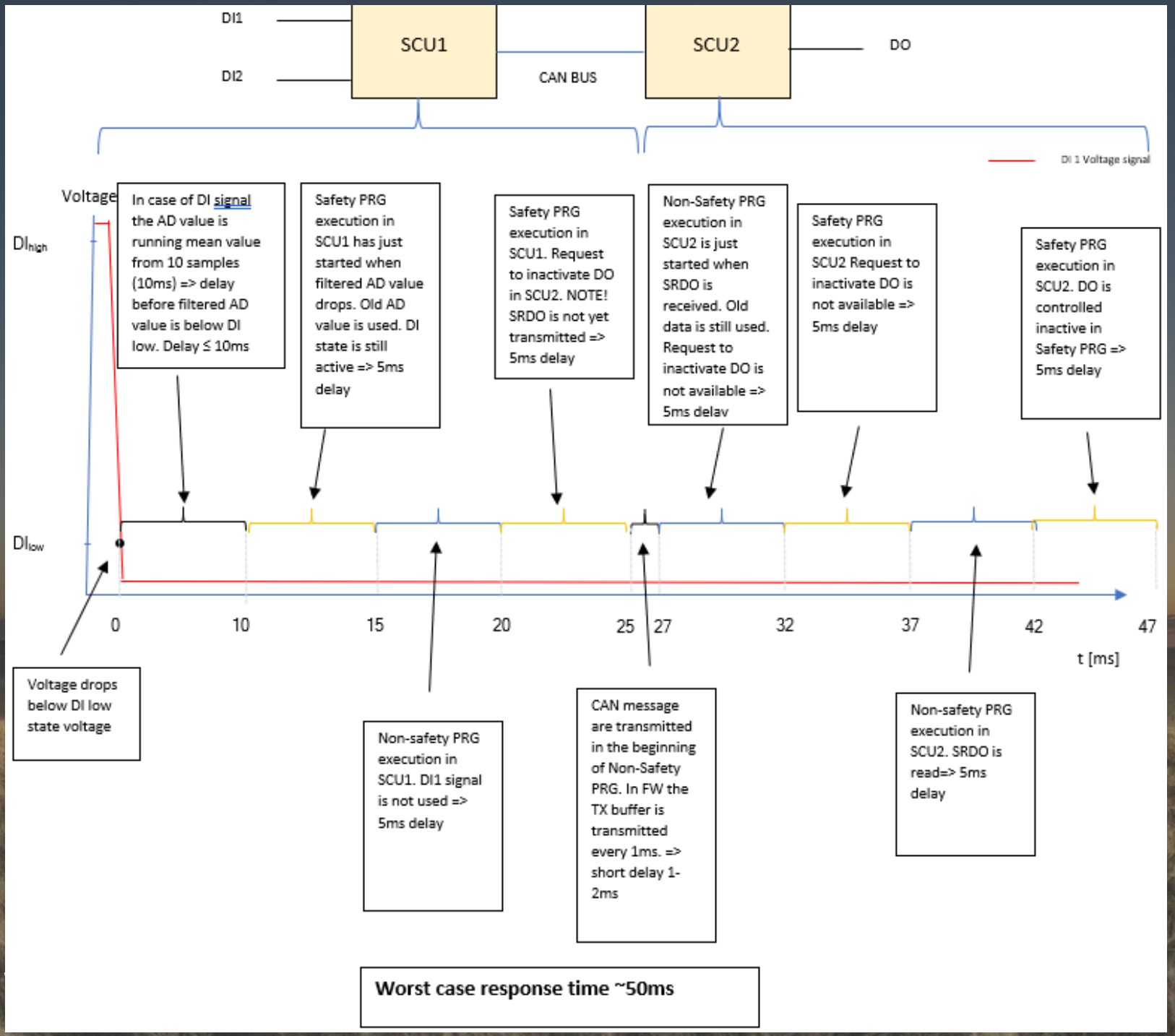
- **Non-safety related POU cannot write to safety related memory area**

- **Safety related POU cannot call non-safety related POU**

RESPONSE TIME – WORST CASE ANALYSIS

- Safety function (a part of a safety related PRG)
  - 2 x DI are read in SCU1 (safety control unit)
    - If either of DI state changes to inactive

      → the DO in SCU2 unit is controlled inactive
  - Both safety units contain a safety PRG and a non-safety PRG (also two tasks)
  - Both tasks have 10 ms cycle time and the execution time of one PRG is 5 ms (in this example)
  - SRDO cycle time 20ms

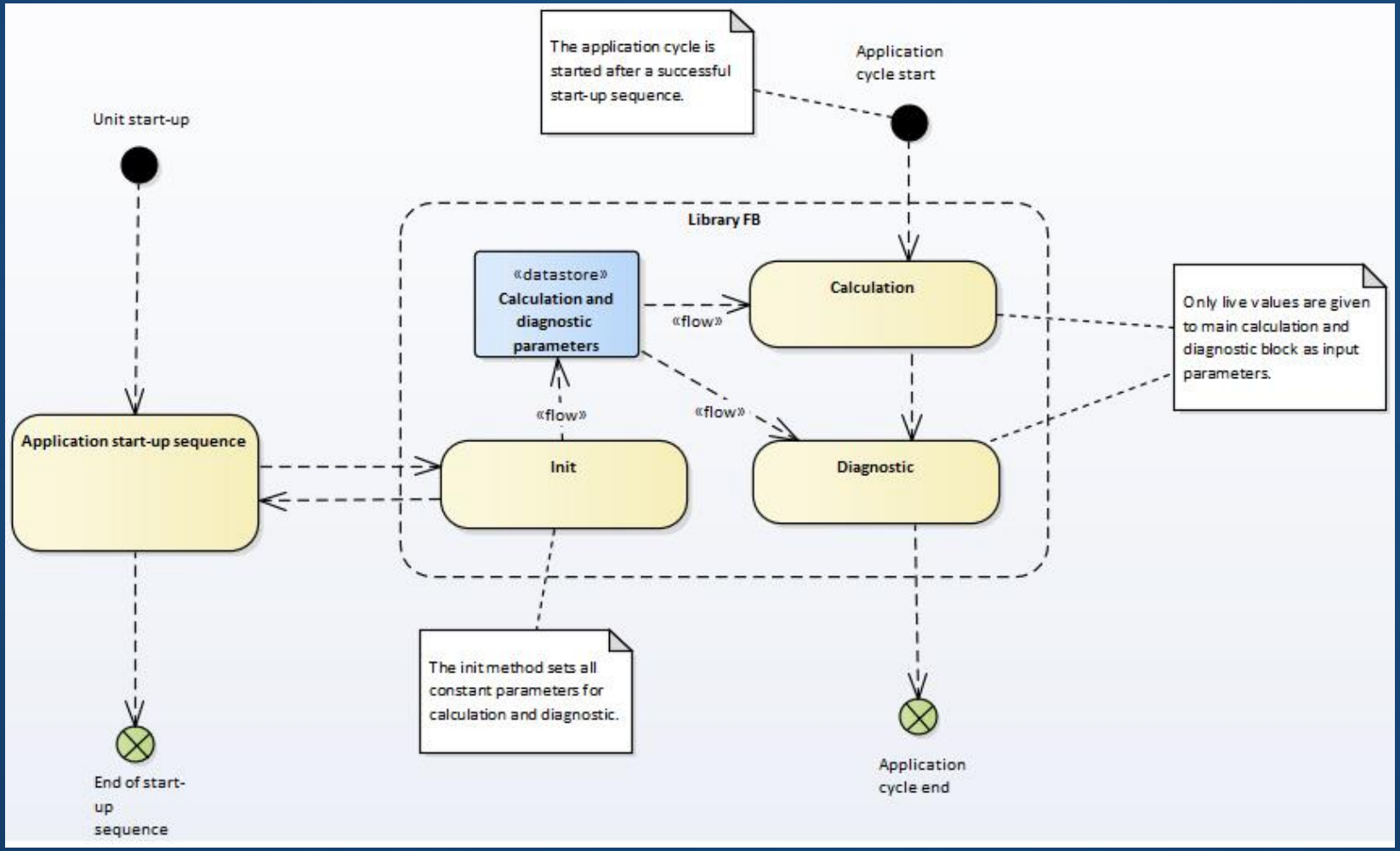

DI1
DI2
SCU1
CAN BUS
SCU2
DO
DI 1 Voltage signal
Voltage
DIhigh
DIlow
In case of DI signal the AD value is running mean value from 10 samples (10ms) => delay before filtered AD value is below DI low. Delay ≤ 10ms
Safety PRG execution in SCU1 has just started when filtered AD value drops. Old AD value is used. DI state is still active => 5ms delay
Safety PRG execution in SCU1. Request to inactivate DO in SCU2. NOTE! SRDO is not yet transmitted => 5ms delay
Non-Safety PRG execution in SCU2 is just started when SRDO is received. Old data is still used. Request to inactivate DO is not available => 5ms delay
Safety PRG execution in SCU2 Request to inactivate DO is not available => 5ms delay
Safety PRG execution in SCU2. DO is controlled inactive in Safety PRG => 5ms delay
0
10
15
20
25
27
32
37
42
47
t [ms]
Voltage drops below DI low state voltage
Non-safety PRG execution in SCU1. DI1 signal is not used => 5ms delay
CAN message are transmitted in the beginning of Non-Safety PRG. In FW the TX buffer is transmitted every 1ms. => short delay 1-2ms
Non-safety PRG execution in SCU2. SRDO is read=> 5ms delay
Worst case response time ~50ms

EPEC

SAFETY RELATED LIBRARY POU

- All safety library function blocks have initialization method
  - called once in program initialization
  - sets all constant parameters for function blocks
    - This reduces the amount of inputs at program cycle calls
- Each safety library has its own set of error statuses defined in public global status structure named *Status*
- When a POU detects any error, any output variables that are SAFE variable type are set to the given safe value/state
  - For function block instances, the safe value is given in the *Init* method
  - If a safe value/state is not given, value zero/state FALSE is used

SAFETY RELATED LIBRARY POU



EPEC

SAFETY RELATED LIBRARIES

- *SafeCANopenSRDO*
  - POUs for creating and validating SRDO messages according to EN50325-5
- *SafeConversion*
  - Conversions from
    - ADC to voltage/current
    - voltage to DI state
    - voltage to resistance
- *SafeDataValidation*
  - Several different data validation operations, for example, SRDO signature calculation/validation, parameter CRC calculation/validation and AI/DI signal validation

SAFETY RELATED LIBRARIES

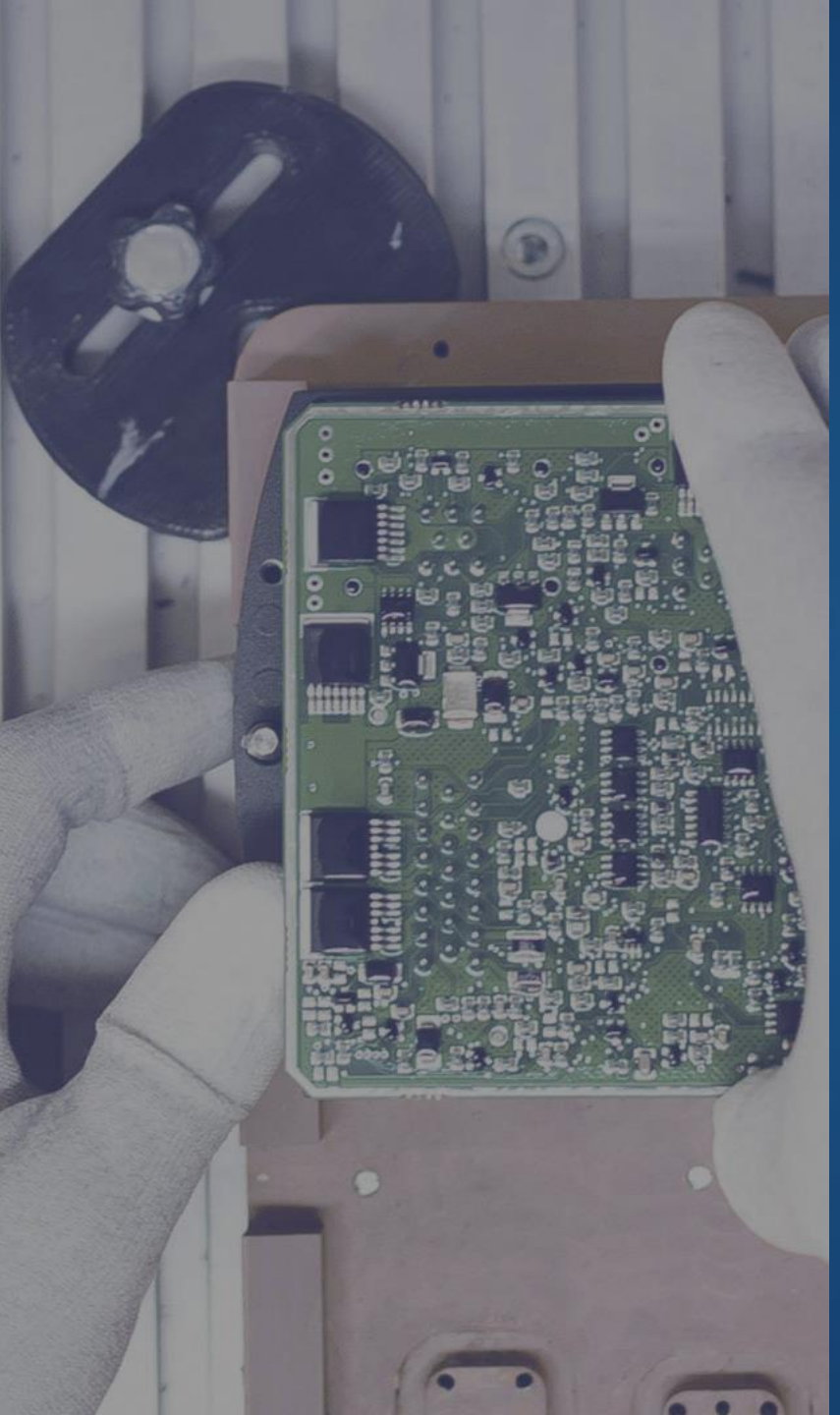
- *SafeJoystickCalibrationAndDiagnostic*
  - POUs for joystick and pedal, including calibration, progression and deadband
- SafeProportionalValveControl
  - For current and voltage controlled valves
  - Improved, adaptive controller
- *SafeSensorCalibration*
  - POUs for KTY and linear sensors
- *DiagnosticInterface*
  - Event code related data types common to all safety libraries

S SERIES LIBRARIES

- *SafeSSeriesHardware*
  - POUs for diagnosing S series hardware voltages, currents and temperature
- *SSeriesHardware*
  - S-Series platform specific non-safe POUs, for example, *NVMem* and *DiagnosticLED*
  - FB for time-triggered CAN messages (*CanApiTimeTriggeredMsg* **)**
- *SafeSC52Int*
  - For example, functions to read and write system parameters
- *SSeriesSystemExt*
  - Functions for handling LED, password, system log, random numbers and system parameters
- *SafeSSeriesIODriverExt*
  - Functions for I/O configuration

COMMON NON-SAFETY LIBRARIES

- *ParameterHandler*
  - Used for parameter image handling
- *CANVXD_API / CANL2_CANVXD*
  - Interface and implementation to use CAN
- *CANopen (v 4.0)*
  - Includes functionality from previously used libraries CANopen, CANopenCSDO, CANopen_302, NetworkManager
- *CANopenODSave (v 4.0)*
  - Provides services to store CANopen Object Dictionary



COMMON NON-SAFETY LIBRARIES

- *DataTransfer*
- *J1939*
  - J1939 protocol related functions
- *EventLog, EventLogTransfer*
  - Event handling functions
- *J1939Event*
  - J1939 related event conversion functions

EPEC

EPEC DOCUMENTATION

- **Epec Extranet, *Programming Manuals***
  - CODESYS Safety SIL2 User manual (PDF)
  - SC52 Safety Manual (PDF)
  - Programming and Libraries Manual (HTML, CHM)
- ***Epec Programming And Libraries Manual***
  - SDK installs to *C:\Program Files (x86)\Epec\SDKDocumentation*
    - Open via **MultiTool > Help**
  - *Programming book*
    - *Programming Safety Projects*
    - *Programming SC52 Safety Control Unit*
  - *Libraries book*
    - *S Series Specific Libraries*
    - *Common Libraries for Safety Project*

Thank you!

Any questions?

Contact our technical support
techsupport@epec.fi