LAr OPC-UA software discussion

Olivier Arnaez

17h of October 2025

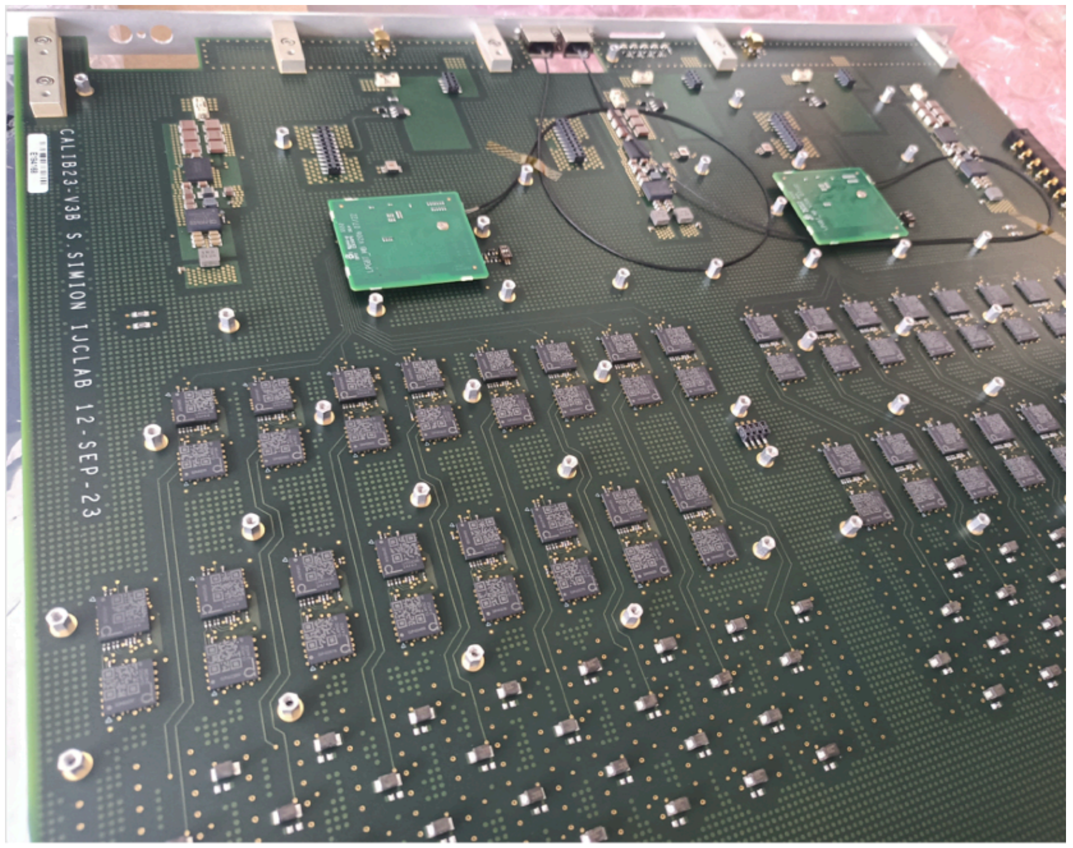
Introduction

- OPC-UA server based on the quasar framework serves as
  - single pipeline for communication with the hardware (to avoid interferences) for both DCS and DAQ paths
  - abstraction of the hardware in order to « hide » all technicalities related to the hardware unnecessary to operate the hardware
    - full access to configuration/monitoring remains possible through low-level functions but are assumed to be not common operations

- Compilation can be done with the free open62541-compat library but eventual operation with the licensed ua-sdk library offering multi-threading
  - open62541-compat could probably be complemented with multi-threading capability but out of scope of my/our work (—> quasar development)

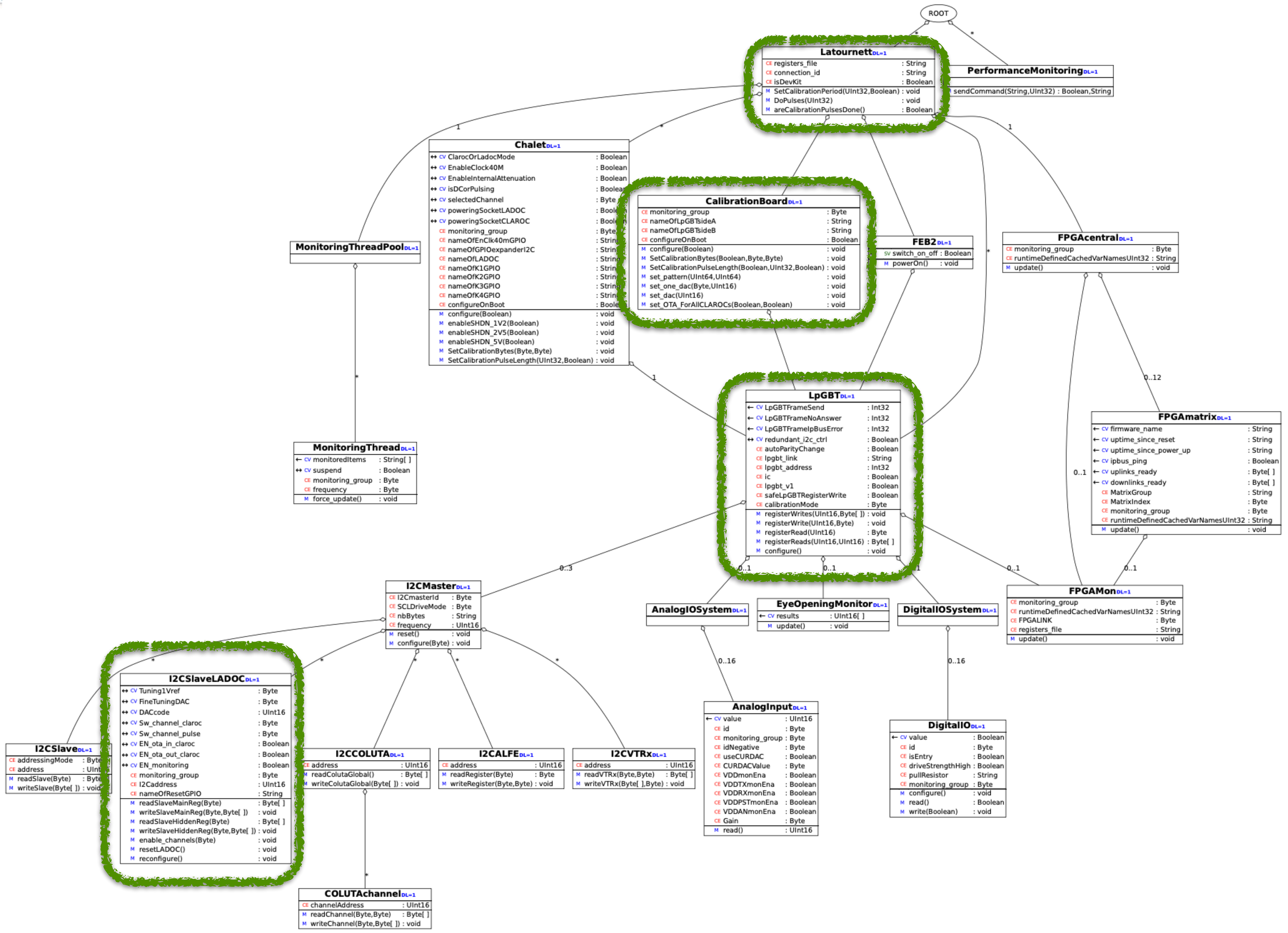
About the calibration side

- Calibration board ASICs quite simple with only a few registers for which convenient accessor functions (and OPC-UA variables/methods) facilitate the reading/writing through relevant naming (e.g. « FineTuningDAC » instead of writeSlaveHiddenRegister(reg=3,…)
- Configuration can be done from values in the server configuration XML file and/or using the OPC-UA methods
  - The former is personally my preferred option since the configuration of ASICs is assumed not to change after installation
- Calibration board's level methods to apply usual common/practical operations to all on-board ASICs (set_dac, set_OTA_ForAllCLAROCs, set_pattern, configure) in order to avoid 32 transactions to configure a full board (loop over ASICs internal to server, optimising the parallelisation)



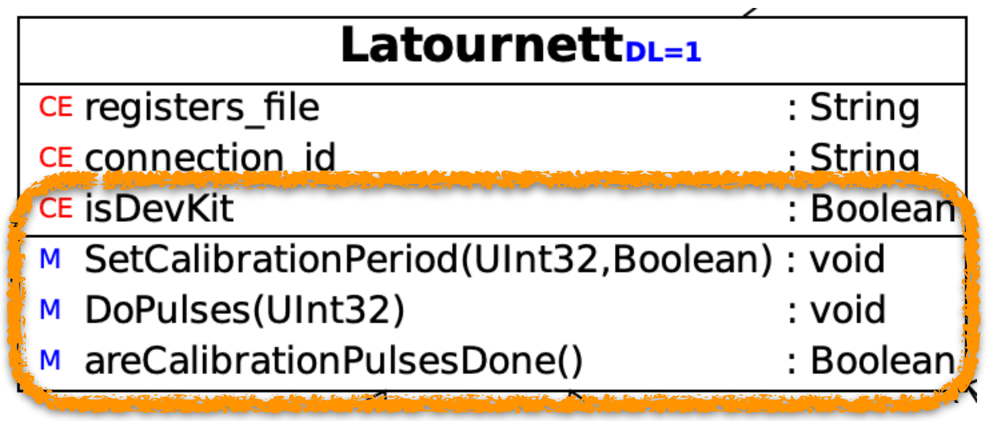
- Full calibration board configuration currently done in ~4.5 seconds
- Some « interface specifications » at here

OPC-UA server architecture



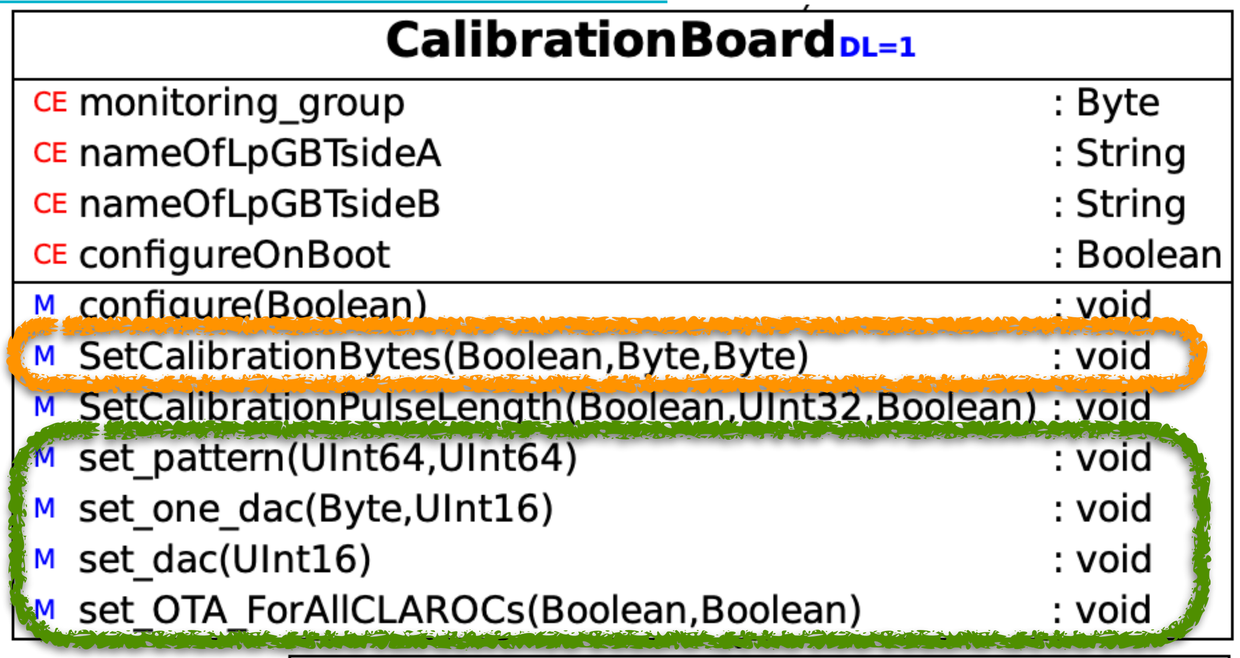
ROOT
*
*
Latournett DL=1
CE registers_file : String
CE connection_id : String
CE isDevKit : Boolean
M SetCalibrationPeriod(UInt32,Boolean) : void
M DoPulses(UInt32) : void
M areCalibrationPulsesDone() : Boolean
PerformanceMonitoring DL=1
M sendCommand(String,UInt32) : Boolean,String
1
*
1
Chalet DL=1
↔ CV ClarocOrLadocMode : Boolean
↔ CV EnableClock40M : Boolean
↔ CV EnableInternalAttenuation : Boolean
↔ CV isDCorPulsing : Boolean
↔ CV selectedChannel : Byte
↔ CV poweringSocketLADOC : Boolean
↔ CV poweringSocketCLAROC : Boolean
CE monitoring_group : Byte
CE nameOfEnClk40mGPIO : String
CE nameOfGPIOexpanderI2C : String
CE nameOfLADOC : String
CE nameOfK1GPIO : String
CE nameOfK2GPIO : String
CE nameOfK3GPIO : String
CE nameOfK4GPIO : String
CE configureOnBoot : Boolean
M configure(Boolean) : void
M enableSHDN_1V2(Boolean) : void
M enableSHDN_2V5(Boolean) : void
M enableSHDN_5V(Boolean) : void
M SetCalibrationBytes(Byte,Byte) : void
M SetCalibrationPulseLength(UInt32,Boolean) : void
CalibrationBoard DL=1
CE monitoring_group : Byte
CE nameOfLpGBTsideA : String
CE nameOfLpGBTsideB : String
CE configureOnBoot : Boolean
M configure(Boolean) : void
M SetCalibrationBytes(Boolean,Byte,Byte) : void
M SetCalibrationPulseLength(Boolean,UInt32,Boolean) : void
M set_pattern(UInt64,UInt64) : void
M set_one_dac(Byte,UInt16) : void
M set_dac(UInt16) : void
M set_OTA_ForAllCLAROCs(Boolean,Boolean) : void
FEB2 DL=1
SV switch_on_off : Boolean
M powerOn() : void
*
FPGAcentral DL=1
CE monitoring_group : Byte
CE runtimeDefinedCachedVarNamesUInt32 : String
M update() : void
MonitoringThreadPool DL=1
*
MonitoringThread DL=1
← CV monitoredItems : String[ ]
↔ CV suspend : Boolean
CE monitoring_group : Byte
CE frequency : Byte
M force_update() : void
1
LpGBT DL=1
← CV LpGBTFrameSend : Int32
← CV LpGBTFrameNoAnswer : Int32
← CV LpGBTFrameIpBusError : Int32
↔ CV redundant_i2c_ctrl : Boolean
CE autoParityChange : Boolean
CE lpgbt_link : String
CE lpgbt_address : Int32
CE ic : Boolean
CE lpgbt_v1 : Boolean
CE safeLpGBTRegisterWrite : Boolean
CE calibrationMode : Byte
M registerWrites(UInt16,Byte[ ]) : void
M registerWrite(UInt16,Byte) : void
M registerRead(UInt16) : Byte
M registerReads(UInt16,UInt16) : Byte[ ]
M configure() : void
0..12
FPGAmatrix DL=1
← CV firmware_name : String
← CV uptime_since_reset : String
← CV uptime_since_power_up : String
← CV ipbus_ping : Boolean
← CV uplinks_ready : Byte[ ]
← CV downlinks_ready : Byte[ ]
CE MatrixGroup : String
CE MatrixIndex : Byte
CE monitoring_group : Byte
CE runtimeDefinedCachedVarNamesUInt32 : String
M update() : void
0..1
0..3
0..1
0..1
1
0..1
0..1
I2CMaster DL=1
CE I2CmasterId : Byte
CE SCLDriveMode : Byte
CE nbBytes : String
CE frequency : UInt16
M reset() : void
M configure(Byte) : void
AnalogIOSystem DL=1
EyeOpeningMonitor DL=1
← CV results : UInt16[ ]
M update() : void
DigitalIOSystem DL=1
FPGAMon DL=1
CE monitoring_group : Byte
CE runtimeDefinedCachedVarNamesUInt32 : String
CE FPGALINK : Byte
CE registers_file : String
M update() : void
*
*
*
*
*
0..16
0..16
I2CSlaveLADOC DL=1
↔ CV Tuning1Vref : Byte
↔ CV FineTuningDAC : Byte
↔ CV DACcode : UInt16
↔ CV Sw_channel_claroc : Byte
↔ CV Sw_channel_pulse : Byte
↔ CV EN_ota_in_claroc : Boolean
↔ CV EN_ota_out_claroc : Boolean
↔ CV EN_monitoring : Boolean
CE monitoring_group : Byte
CE I2Caddress : UInt16
CE nameOfResetGPIO : String
M readSlaveMainReg(Byte) : Byte[ ]
M writeSlaveMainReg(Byte,Byte[ ]) : void
M readSlaveHiddenReg(Byte) : Byte[ ]
M writeSlaveHiddenReg(Byte,Byte[ ]) : void
M enable_channels(Byte) : void
M resetLADOC() : void
M reconfigure() : void
I2CSlave DL=1
CE addressingMode : Byte
CE address : UInt16
M readSlave(Byte) : Byte
M writeSlave(Byte[ ]) : void
I2CCOLUTA DL=1
CE address : UInt16
M readColutaGlobal() : Byte[ ]
M writeColutaGlobal(Byte[ ]) : void
I2CALFE DL=1
CE address : UInt16
M readRegister(Byte) : Byte
M writeRegister(Byte,Byte) : void
I2CVTRx DL=1
CE address : UInt16
M readVTRx(Byte,Byte) : Byte[ ]
M writeVTRx(Byte[ ],Byte) : void
AnalogInput DL=1
← CV value : UInt16
CE id : Byte
CE monitoring_group : Byte
CE idNegative : Byte
CE useCURDAC : Boolean
CE CURDACValue : Byte
CE VDDmonEna : Boolean
CE VDDTXmonEna : Boolean
CE VDDRXmonEna : Boolean
CE VDDPSTmonEna : Boolean
CE VDDANmonEna : Boolean
CE Gain : Byte
M read() : UInt16
DigitalIO DL=1
← CV value : Boolean
CE id : Byte
CE isEntry : Boolean
CE driveStrengthHigh : Boolean
CE pullResistor : String
CE monitoring_group : Byte
M configure() : void
M read() : Boolean
M write(Boolean) : void
*
COLUTAchannel DL=1
CE channelAddress : UInt16
M readChannel(Byte,Byte) : Byte[ ]
M writeChannel(Byte,Byte[ ]) : void

Calibration-related aspects

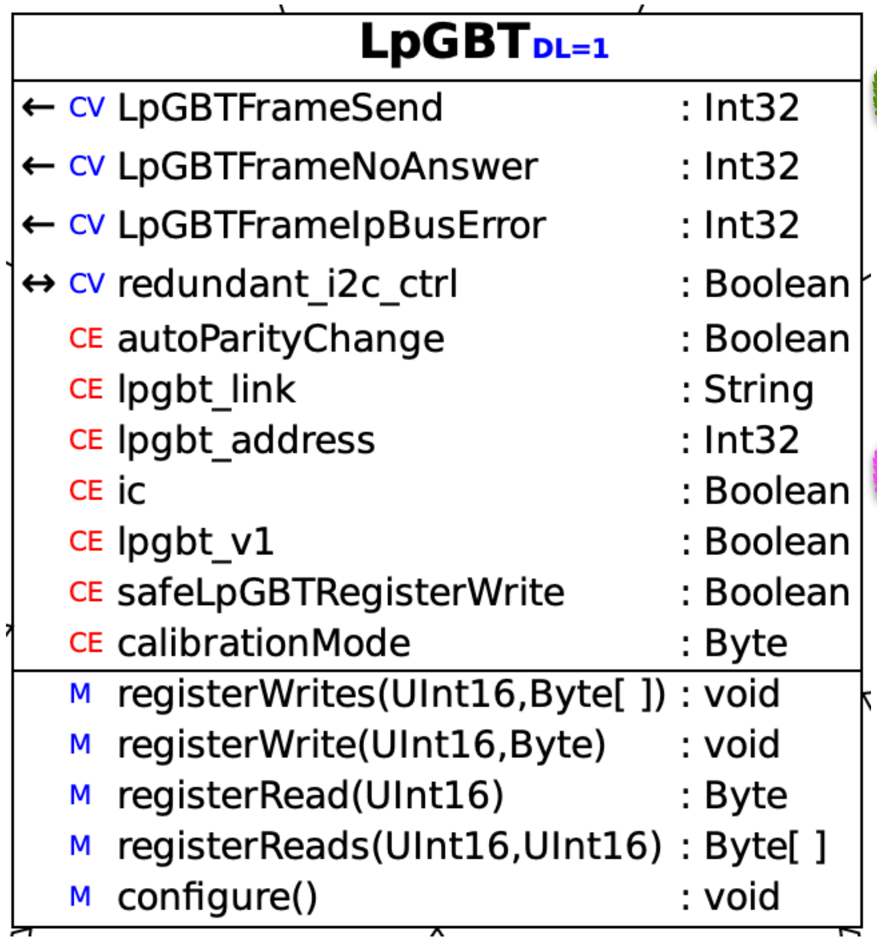


```
Latournett DL=1
CE registers_file                       : String
CE connection_id                        : String
CE isDevKit                             : Boolean
M  SetCalibrationPeriod(UInt32,Boolean) : void
M  DoPulses(UInt32)                     : void
M  areCalibrationPulsesDone()           : Boolean
```

🟧 Used only with internal pulse generator (not « nominal » operation)
Will be renamed/slightly modified to be more elegant



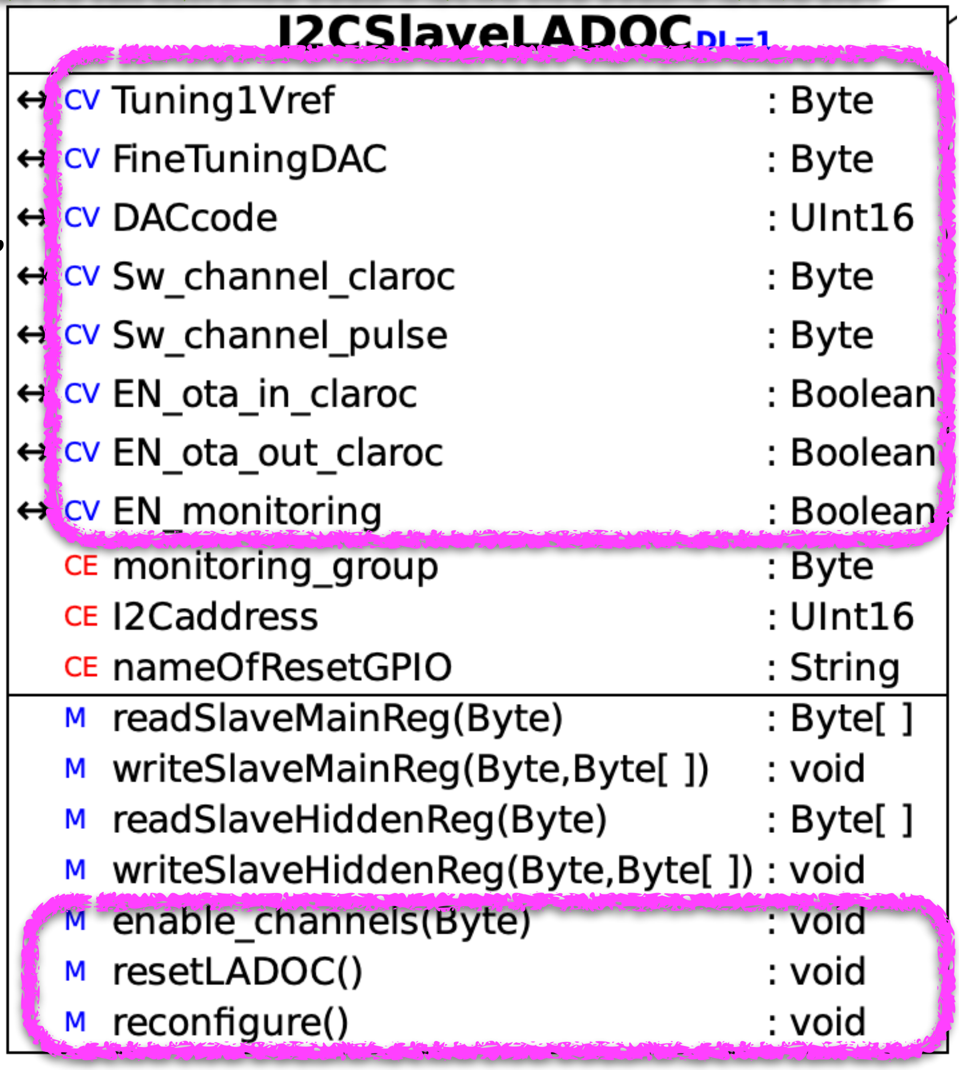
```
CalibrationBoard DL=1
CE monitoring_group                                     : Byte
CE nameOfLpGBTsideA                                     : String
CE nameOfLpGBTsideB                                     : String
CE configureOnBoot                                      : Boolean
M  configure(Boolean)                                   : void
M  SetCalibrationBytes(Boolean,Byte,Byte)               : void
M  SetCalibrationPulseLength(Boolean,UInt32,Boolean)    : void
M  set_pattern(UInt64,UInt64)                           : void
M  set_one_dac(Byte,UInt16)                             : void
M  set_dac(UInt16)                                      : void
M  set_OTA_ForAllCLAROCs(Boolean,Boolean)               : void
```



```
LpGBT DL=1
← CV LpGBTFrameSend                 : Int32
← CV LpGBTFrameNoAnswer             : Int32
← CV LpGBTFrameIpBusError           : Int32
↔ CV redundant_i2c_ctrl             : Boolean
  CE autoParityChange               : Boolean
  CE lpgbt_link                     : String
  CE lpgbt_address                  : Int32
  CE ic                             : Boolean
  CE lpgbt_v1                       : Boolean
  CE safeLpGBTRegisterWrite         : Boolean
  CE calibrationMode                : Byte
  M  registerWrites(UInt16,Byte[ ]) : void
  M  registerWrite(UInt16,Byte)     : void
  M  registerRead(UInt16)           : Byte
  M  registerReads(UInt16,UInt16)   : Byte[ ]
  M  configure()                    : void
```

🟩 Calibration board's level methods to address all underlying chips at once (single OPC-UA transaction and internal server optimisations of ipBus, lpGBT and I2C transactions)

🟪 ASIC-level configuration/reset methods



```
I2CSlaveLADOC DL=1
↔ CV Tuning1Vref                          : Byte
↔ CV FineTuningDAC                        : Byte
↔ CV DACcode                              : UInt16
↔ CV Sw_channel_claroc                    : Byte
↔ CV Sw_channel_pulse                     : Byte
↔ CV EN_ota_in_claroc                     : Boolean
↔ CV EN_ota_out_claroc                    : Boolean
↔ CV EN_monitoring                        : Boolean
  CE monitoring_group                     : Byte
  CE I2Caddress                           : UInt16
  CE nameOfResetGPIO                      : String
  M  readSlaveMainReg(Byte)               : Byte[ ]
  M  writeSlaveMainReg(Byte,Byte[ ])      : void
  M  readSlaveHiddenReg(Byte)             : Byte[ ]
  M  writeSlaveHiddenReg(Byte,Byte[ ])    : void
  M  enable_channels(Byte)                : void
  M  resetLADOC()                         : void
  M  reconfigure()                        : void
```

Probably not needed to play with lpGBT on DAQ side (except for checking that it answers but need common with DCS so info could be just provided as an OPC-UA node)