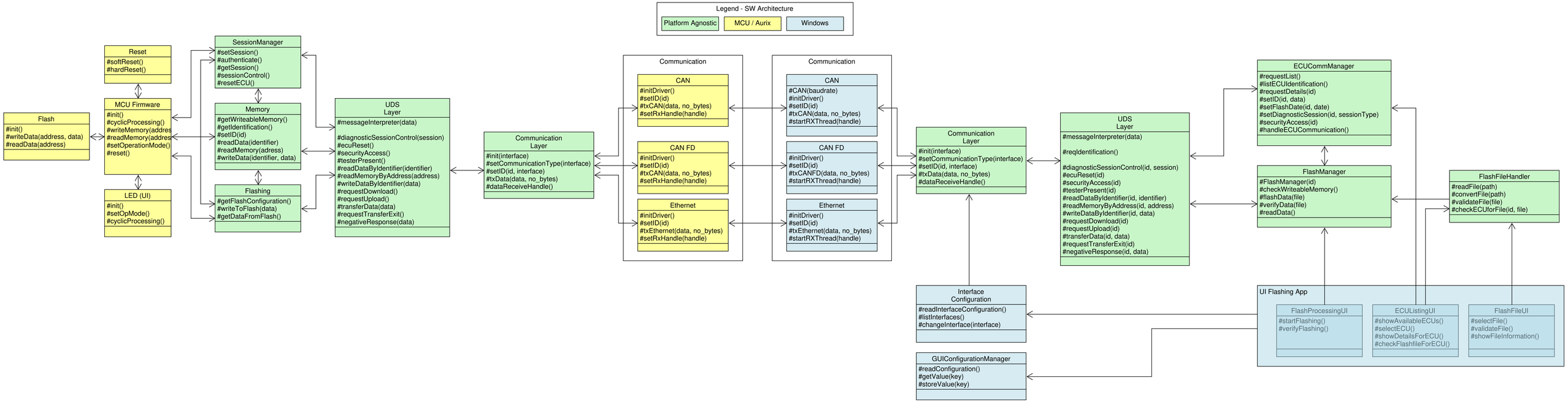
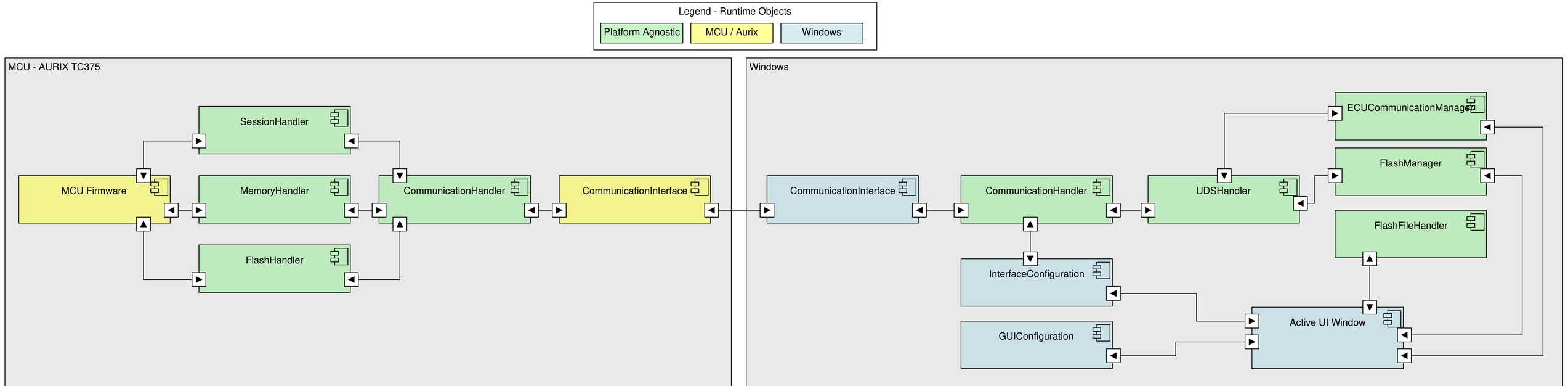

Legend - SW Architecture
Platform Agnostic
MCU / Aurix
Windows
Flash
#init()
#writeData(address, data)
#readData(address)
Reset
#softReset()
#hardReset()
MCU Firmware
#init()
#cyclicProcessing()
#writeMemory(addres
#readMemory(addres
#setOperationMode()
#reset()
LED (UI)
#init()
#setOpMode()
#cyclicProcessing()
SessionManager
#setSession()
#authenticate()
#getSession()
#sessionControl()
#resetECU()
Memory
#getWriteableMemory()
#getIdentification()
#setID(id)
#readData(identifier)
#readMemory(adress)
#writeData(identifier, data)
Flashing
#getFlashConfiguration()
#writeToFlash(data)
#getDataFromFlash()
UDS
Layer
#messageInterpreter(data)
#diagnosticSessionControl(session)
#ecuReset()
#securityAccess()
#testerPresent()
#readDataByIdentifier(identifier)
#readMemoryByAddress(address)
#writeDataByIdentifier(data)
#requestDownload()
#requestUpload()
#transferData(data)
#requestTransferExit()
#negativeResponse(data)
Communication
Layer
#init(interface)
#setCommunicationType(interface)
#setID(id, interface)
#txData(data, no_bytes)
#dataReceiveHandle()
Communication
CAN
#initDriver()
#setID(id)
#txCAN(data, no_bytes)
#setRxHandle(handle)
CAN FD
#initDriver()
#setID(id)
#txCAN(data, no_bytes)
#setRxHandle(handle)
Ethernet
#initDriver()
#setID(id)
#txEthernet(data, no_bytes)
#setRxHandle(handle)
Communication
CAN
#CAN(baudrate)
#initDriver()
#setID(id)
#txCAN(data, no_bytes)
#startRXThread(handle)
CAN FD
#initDriver()
#setID(id)
#txCANFD(data, no_bytes)
#startRXThread(handle)
Ethernet
#initDriver()
#setID(id)
#txEthernet(data, no_bytes)
#startRXThread(handle)
Communication
Layer
#init(interface)
#setCommunicationType(interface)
#setID(id, interface)
#txData(data, no_bytes)
#dataReceiveHandle()
UDS
Layer
#messageInterpreter(data)
#reqIdentification()
#diagnosticSessionControl(id, session)
#ecuReset(id)
#securityAccess(id)
#testerPresent(id)
#readDataByIdentifier(id, identifier)
#readMemoryByAddress(id, address)
#writeDataByIdentifier(id, data)
#requestDownload(id)
#requestUpload(id)
#transferData(id, data)
#requestTransferExit(id)
#negativeResponse(id, data)
ECUCommManager
#requestList()
#listECUIdentification()
#requestDetails(id)
#setID(id, data)
#setFlashDate(id, date)
#setDiagnosticSession(id, sessionType)
#securityAccess(id)
#handleECUCommunication()
FlashManager
#FlashManager(id)
#checkWriteableMemory()
#flashData(file)
#verifyData(file)
#readData()
FlashFileHandler
#readFile(path)
#convertFile(path)
#validateFile(file)
#checkECUforFile(id, file)
Interface
Configuration
#readInterfaceConfiguration()
#listInterfaces()
#changeInterface(interface)
GUIConfigurationManager
#readConfiguration()
#getValue(key)
#storeValue(key)
UI Flashing App
FlashProcessingUI
#startFlashing()
#verifyFlashing()
ECUListingUI
#showAvailableECUs()
#selectECU()
#showDetailsForECU()
#checkFlashfileForECU()
FlashFileUI
#selectFile()
#validateFile()
#showFileInformation()


Legend - Runtime Objects
Platform Agnostic
MCU / Aurix
Windows
MCU - AURIX TC375
SessionHandler
MCU Firmware
MemoryHandler
CommunicationHandler
CommunicationInterface
FlashHandler
Windows
CommunicationInterface
CommunicationHandler
UDSHandler
ECUCommunicationManager
FlashManager
FlashFileHandler
InterfaceConfiguration
GUIConfiguration
Active UI Window

Technology Stack

- QT - framework for creating GUI for our application
- QT Creator - development environment for the QT framework
- C++/C - languages used to develop our project
- UDS - communication protocol
- CAN / CAN FD - Can bus for communication
- (Ethernet) - optional - Communication
- Vector CANoe - Debugging/Analyzing messages
- Vector Hardware Manager / Hardware Configuration - Configuration tool for Vector Hardware
- Vector XL Driver Library - Windows driver for Vector hardware
- Aurix - Aurix development studio: IDE
- MinGW - GCC Compiler for Windows
- Eclipse CDT - IDE
- Git - version control system to track the software development process

Diagram Description

Code Components

GUI

UI Flashing App (Windows GUI)

- Selection of files to flash to the microcontroller
- Viewing the ECU and selection for flashing
- Viewing the flash process

Interface Configuration

- User configurations for communication with microcontroller

GUIConfigurationManager

- Handling of configuration file for the GUI

FlashFileHandler

- Reading, converting and validation of flash files

FlashManager

- Responsible for flashing of files to a specific ECU

ECUCommManager

- Handling the communication to ECUs on the bus
- Controlling session and access

MCU

Flash

- interface for flash driver of microcontroller
- erasing and writing in the flash

LED (UI)

- Using LED on the MCU
- Indicate current status of the MCU (connected, flashing etc.)

Reset

- Switch between Bootloader and Application SW

SessionManager

- Handling the session

Memory

- Controlling access to memory

Flashing

- Controlling the flash process for new software

MCU Firmware

- firmware and driver of microcontroller
- Controlling Flash, LED (UI), Reset and Communication

Shared

CAN / CAN FD / Ethernet

- Handling communication for the specific protocol used

UDS Layer

- Standardized communication protocol: 5. and 7. layer of OSI model
- Providing only the necessary UDS services

Communication Layer

- Handling the different interfaces for CAN, CAN FD, Ethernet

Runtime Components

Each component runs as one instance and communicates with the others. The left side shows the MCU runtime containing the bootloader logic and the right side the runtime components in Windows. The Windows UI components seen in the code components are all grouped together as the running Windows UI.