Features and Capabilities of OpenECU

A powerful RCP solution for model-based development

Arnav Gupta
Systems Engineer

May 14th, 2020

DANA OPEN ECU™ EMBEDDED CONTROL

Webinar Information

- Estimated duration: 45 minutes

- Questions:
  - All questions will be answered after the presentation
  - Please feel free to post your questions in the chat window
  - Provide a slide number with your question, if possible

- The presentation will be delivered to you afterwards via email
- Webinar recording will be available on-demand on our website
  - Website: https://www.pi-innovo.com/category/news/papers-and-presentations/
  - Link will be included in an email to you once it becomes available

Presenter

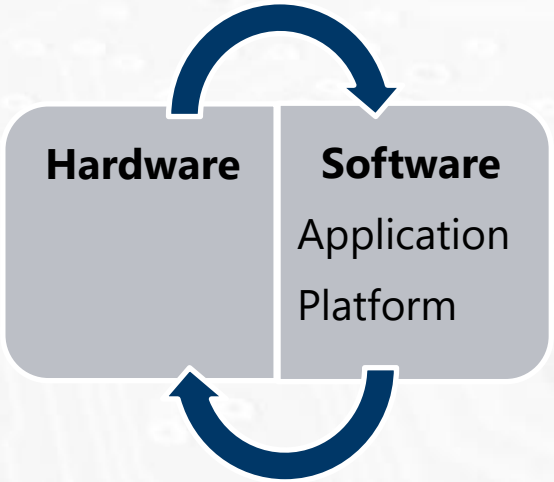
Arnav Gupta
Systems Engineer, OpenECU LLC
arnav.gupta@pi-Innovo.com

Agenda

- MBD and RCP in embedded control software development
- RCP tool evaluation criteria
  - Traditional vs. Embedded
  - Hardware
  - Software
- OpenECU as a proposed on-target RCP solution
  - Architecture
  - Components and modes of ECU
  - Application Software API

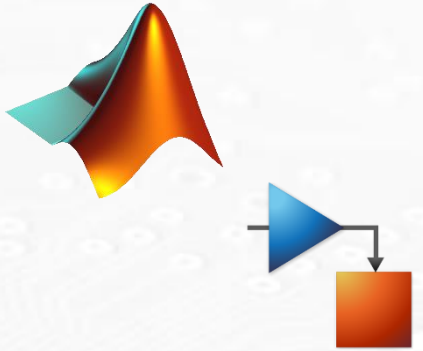
Embedded control software development

- Element in a product bound by hardware, environment, type and properties of the I/O
- Constraints of the system need to be considered
- Encompasses application software and platform software
- Virtual environment is not enough
- Gap between the virtual model and actual hardware needs to be bridged – sooner the better!



Model-based development (MBD)

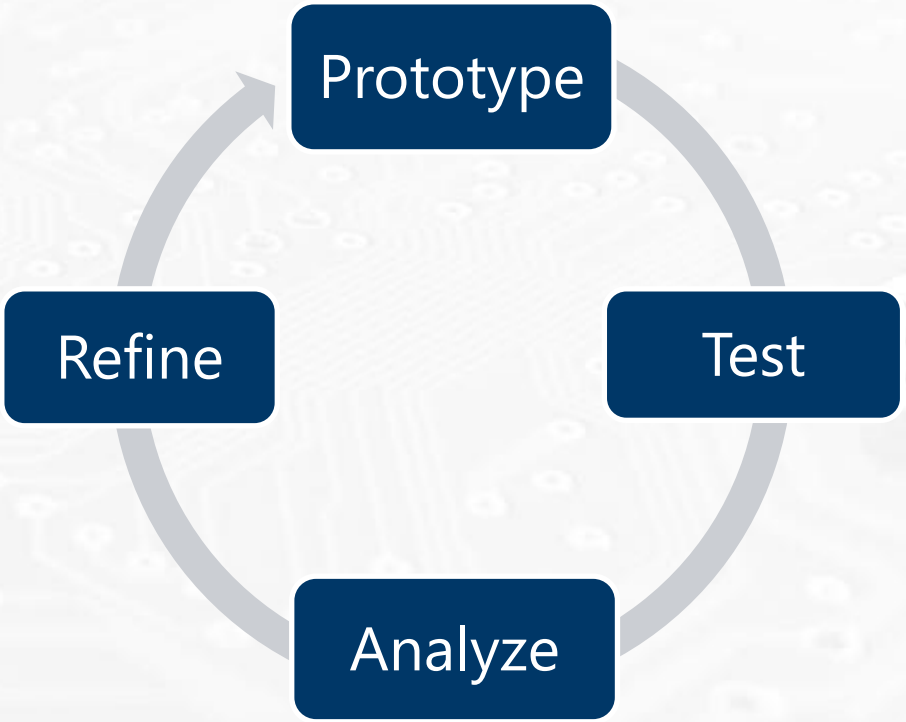
- Environment with graphical and mathematical representation
- Allows early and continuous test and verification
- Automatic code generation – use for rapid prototyping (RP) and production
- Key benefits:
  - Better complexity management
  - Higher development efficiency



Role of RCP in MBD workflow

- Deployment of control algorithms on hardware, in an actual physical environment early in the design process

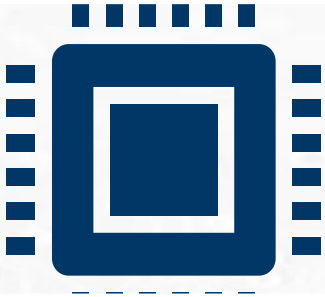
- Key benefits:
  - Early prove out of control functions
  - On-the-fly tuning and testing of software design
  - Allows hard real time operation – key in embedded controls
  - Cost and risk reduction



Traditional vs. Embedded RCP systems

Traditional RCP systems

VS.

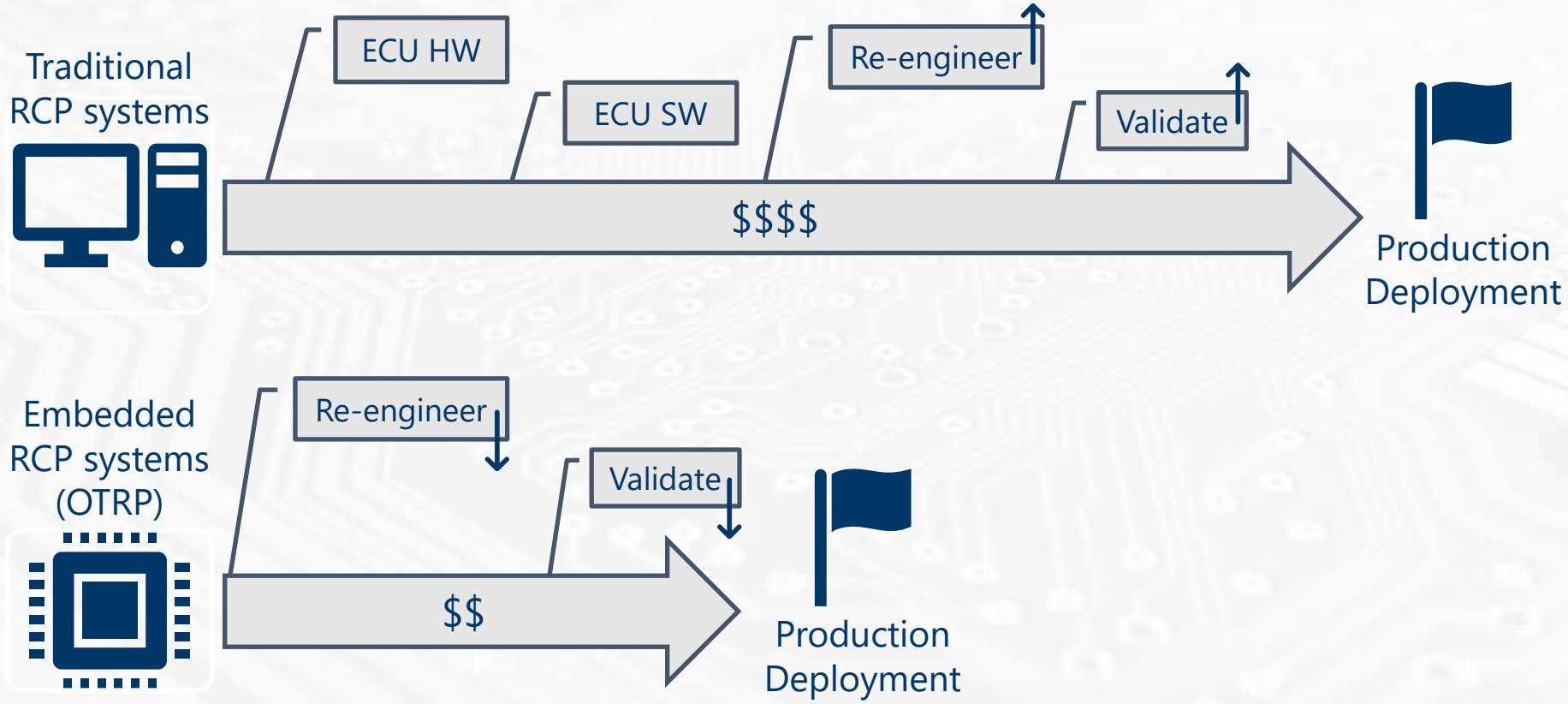


Embedded RCP systems (OTRP)

- Use PC or non-target HW
- Wide variety of I/O available – proprietary configuration and setup
- Expensive and form factor can be an issue downstream
- Output of RP phase might not fit into the resource constrained ECU

- Use ECU or near-production HW
- Use OTS hardware - less expensive, quick setup
- Can be deployed in fleet trials
- Continuous assessment of code within the constraints of an ECU

Traditional vs. Embedded RCP systems



RCP tool evaluation criteria – target HW (ECU)

- Recognize constraints based on cost, size, available processing power, memory, and I/O needs
- Hardware testing
  - HW/SW integration
  - DV
  - Production
  - PV
- Account for
  - Physical interfaces
  - Mechanical interfaces
  - Ingress protection
- Adequate supply chain management
  - Component selection, sourcing and maintenance

DV Test Flow

RCP tool evaluation criteria – platform SW

Low-level software to consider essential supplementary information relevant to target hardware

- Tie the application to the outside world and ECU specific behavior
- Developed using C or assembly language, but integrates with MBD & RCP

Most significant performance and scope impacts come from the software stack outside the application, such as:

- Firmware: bootloader, reset mechanism, ECU modes including reprogramming
- Real time operating system (RTOS), task scheduling
- Memory partitioning and management
- Communication drivers, I/O management
- Low-level device drivers to tie the microprocessor-level details to the application software

Solution for development

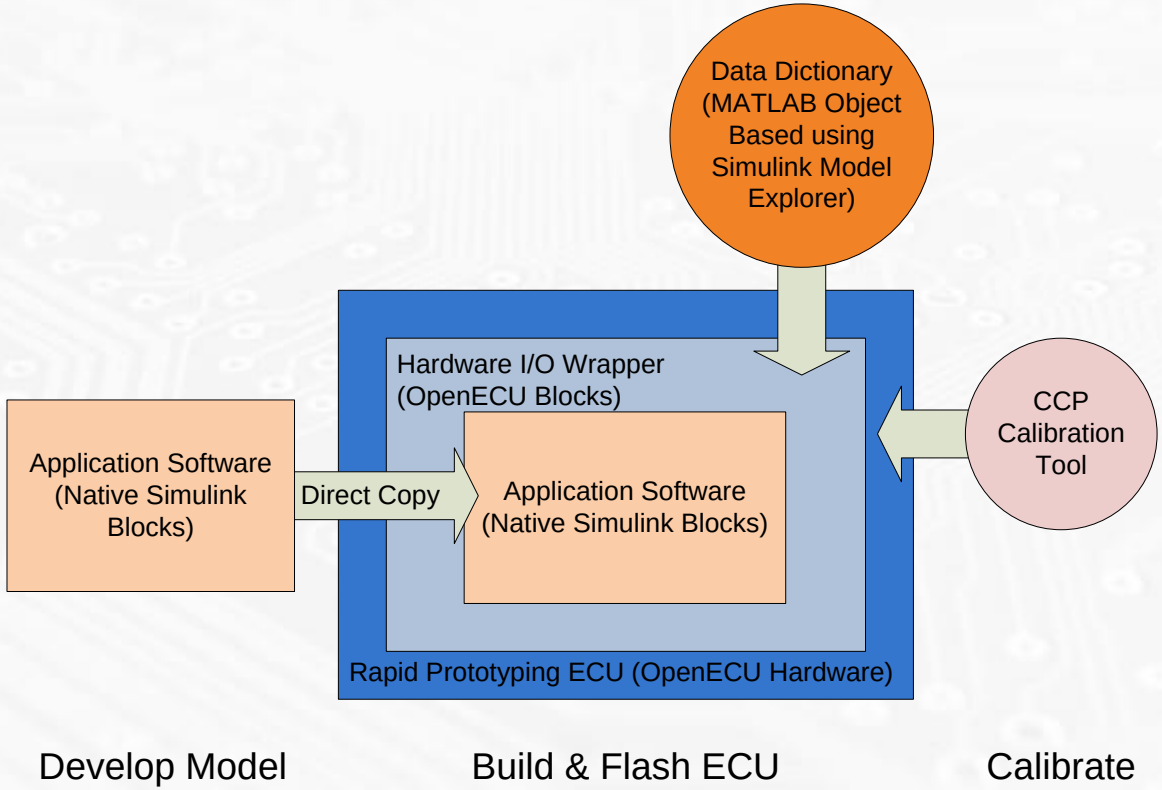
Strong support for MBD

Work at a functional level

Prototyping → Production

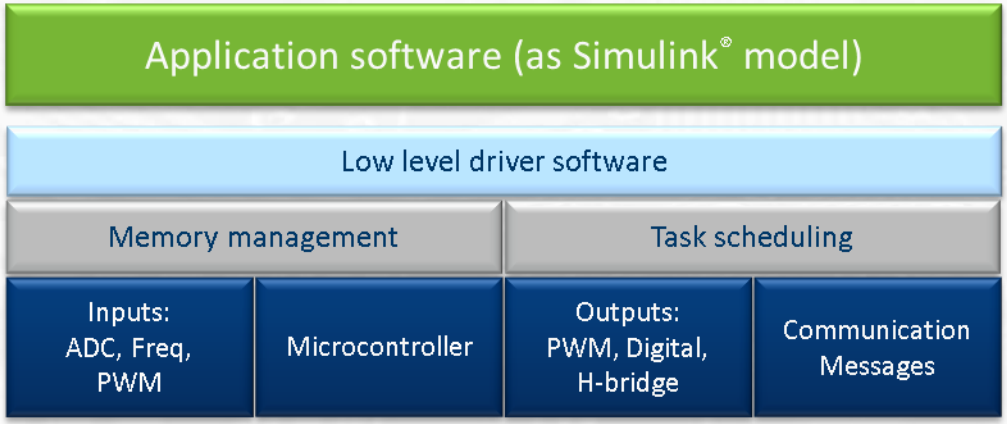
Customizable and reusable HW and SW platforms

Integration with OpenECU solution



OpenECU: abstraction and architecture

- Engineers focus on application specific control system, not platform



Application specific software

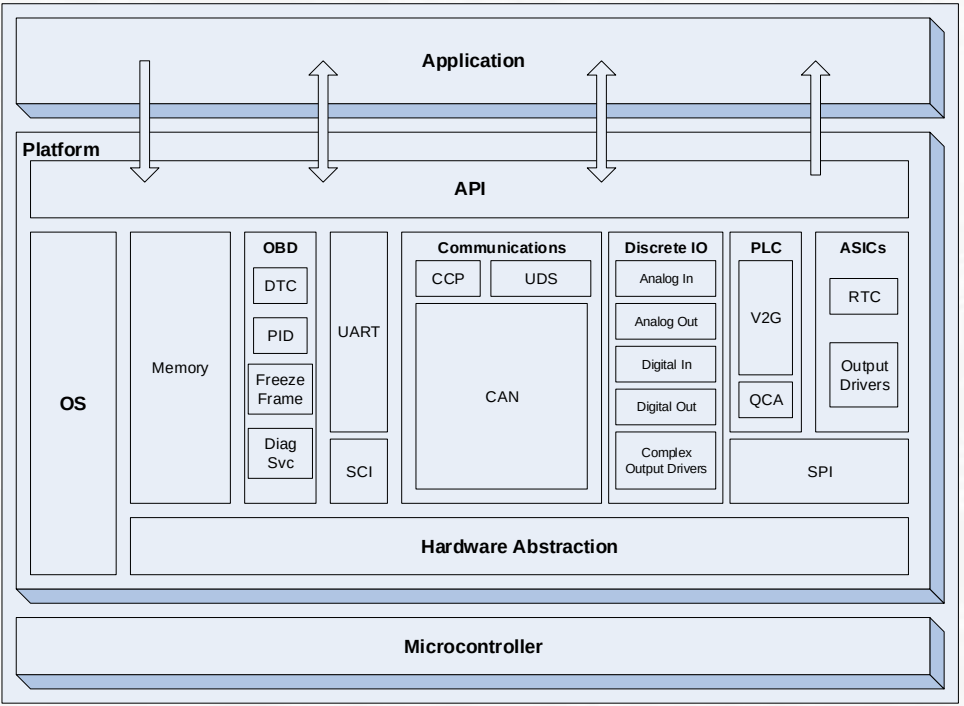
Software

Hardware

Proven OpenECU Platform

- Platform manages complex software & hardware interactions

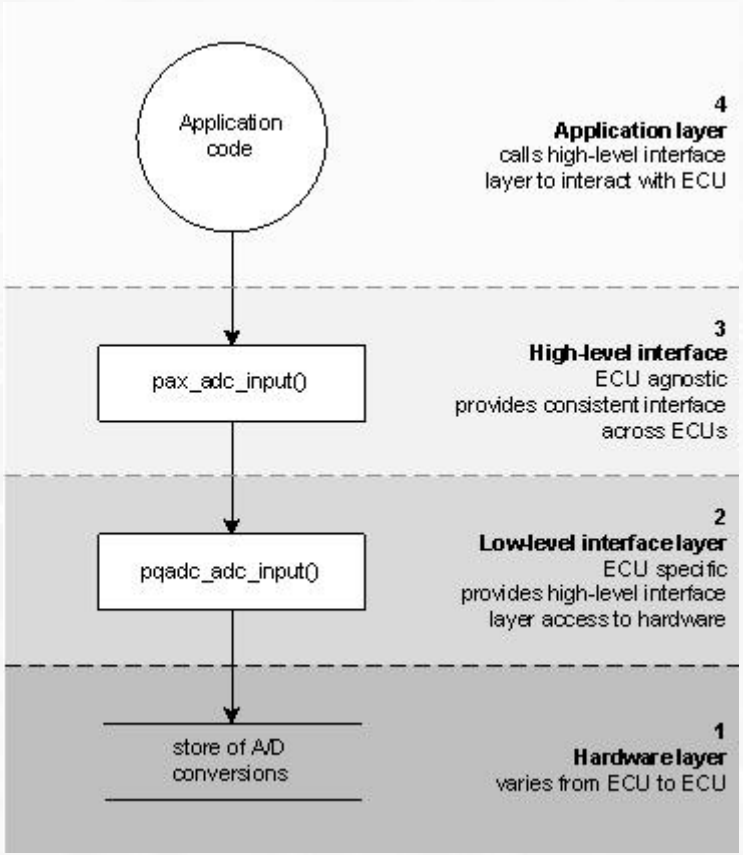
OpenECU platform software architecture



- Real time Operating System (RTOS)
- Memory management
- Communications
- On Board Diagnostics
- Inputs/Outputs
- ASICs
- UART*
- PLC*
- RTC*

*Only on the M560/M580 ECUs

Platform layering example



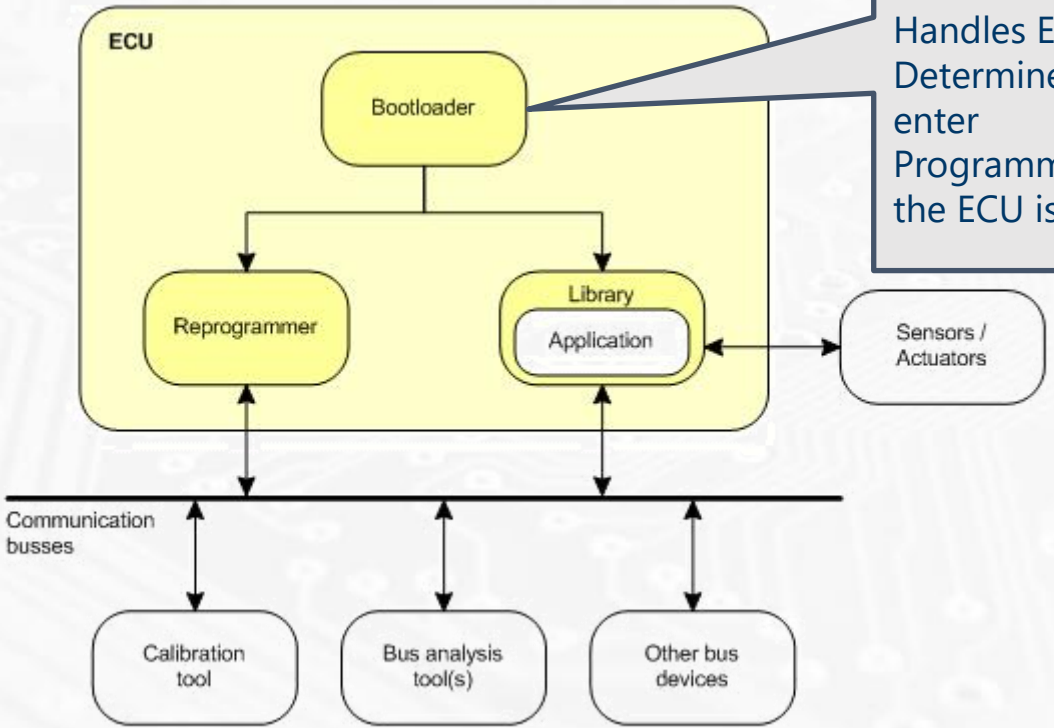
Task scheduling with OpenECU

- RTOS for OpenECU follows a fixed priority pre-emptive scheduling scheme

- Unlike some rapid prototyping environments, subsystems in an OpenECU model do not need to be explicitly triggered
  - Aids model clarity
  - User defines the task rates (Fastest periodic task rate of 1ms possible)

- Starting with the highest priority task, the platform prioritizes execution of tasks as follows: angular, fastest rate task, ..., slowest rate task.

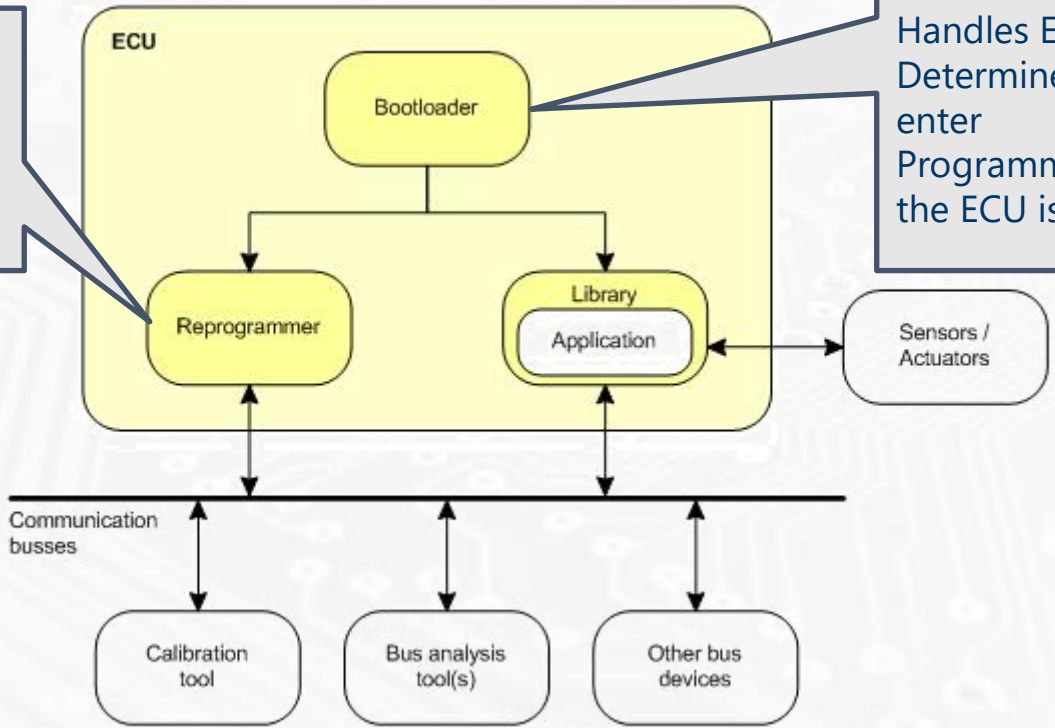
System Components



Handles ECU start-up
Determines the system mode to enter
Programmed into the ECU when the ECU is manufactured

System Components

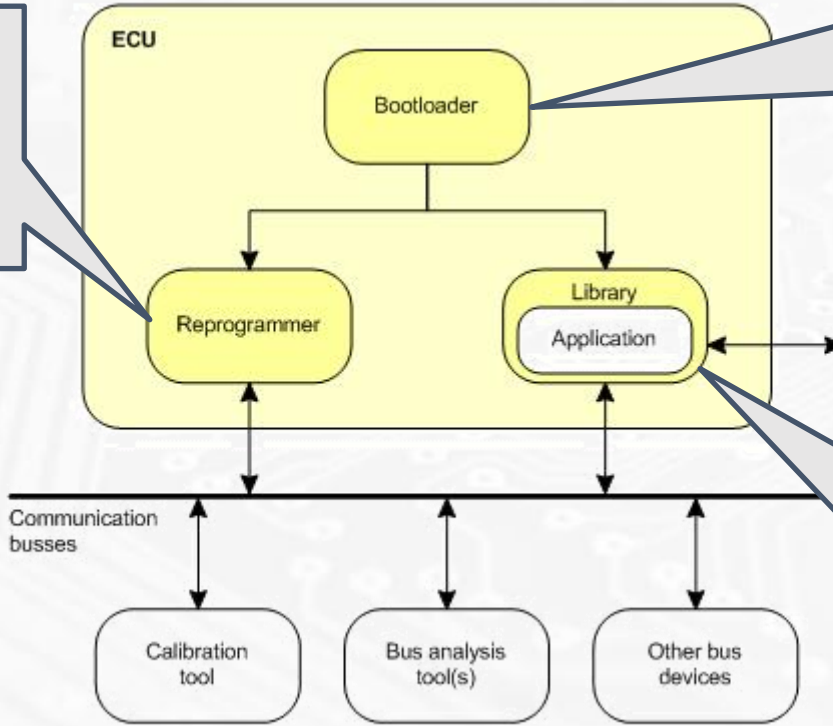
Handles requests to reprogram the application code



Handles ECU start-up
Determines the system mode to enter
Programmed into the ECU when the ECU is manufactured

System Components

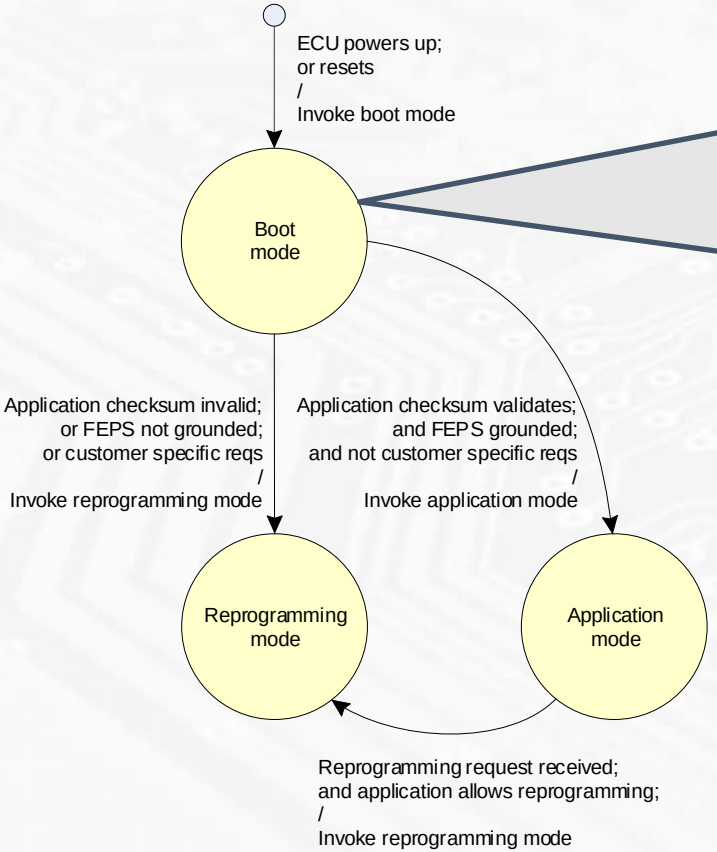
Handles requests to reprogram the application code



Handles ECU start-up
Determines the system mode to enter
Programmed into the ECU when the ECU is manufactured

Library: Provides utilities for start-up, communication, I/O and non-volatile functionality used by the application code
Application: Contains control algorithms, uses library to access sensors, actuators and communication buses

ECU Modes



Boot Mode

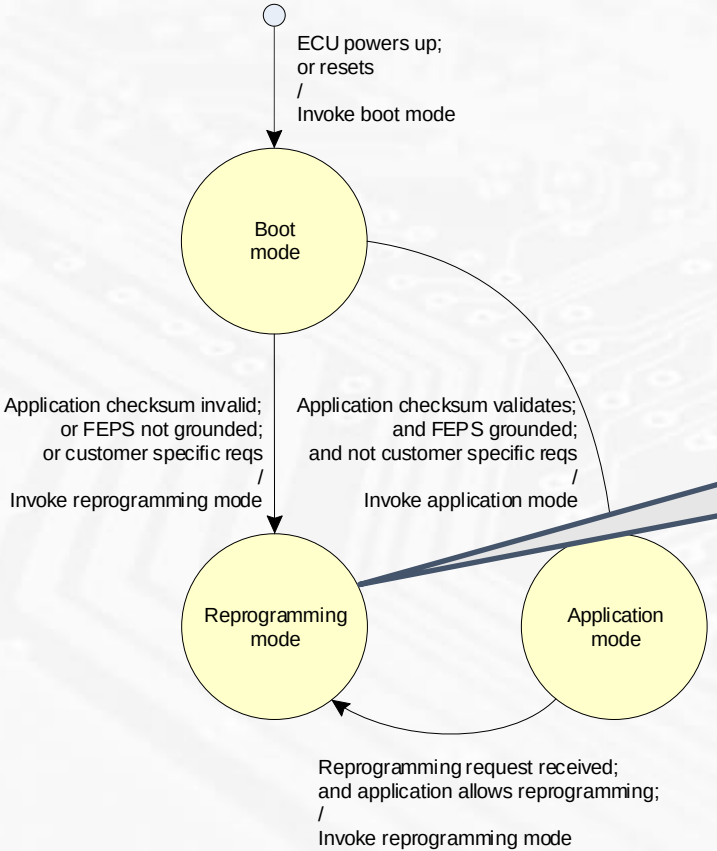
Perform various tests when ECU turned on or recovering from a powered reset:

- Tests on memory devices
- Tests on the code to run
- Tests on the frequency of reset

If tests fail: Either reset or attempt to enter reprogramming mode

If tests pass: Determine what mode to enter next

ECU Modes

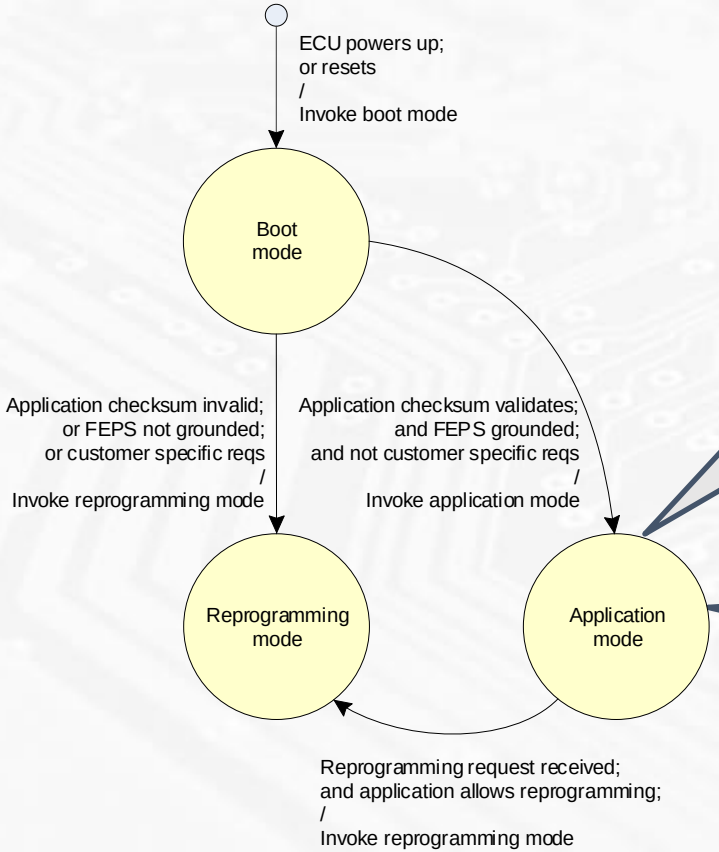


Reprogramming Mode

Allows flashing an application to the ECU. This mode is entered if:

- Flash and EEPROM Programming Signal (FEPS) pin is asserted during power-up
- There is an invalid application image in memory during power-up
- Application allows reprogramming requests via CCP

ECU Modes



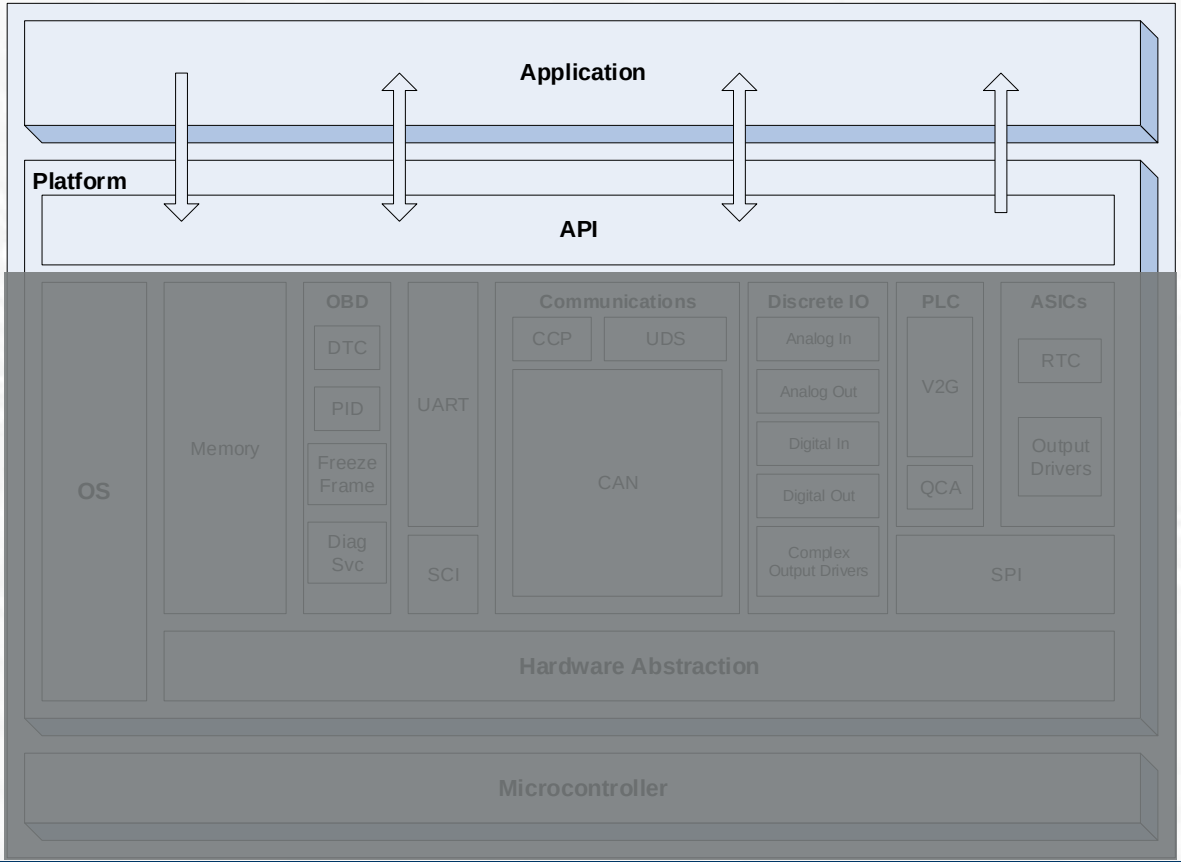
Application Mode

Entered after boot if the FEPS pin is not asserted and a valid application image is present in the memory

Involves the following mechanism:

- Library pre-initialization
- Application initialization
- Library post-initialization
- Application Run

API for application software



API Layer

Interface type	Functionality
Input Drivers	PAI – analog inputs PDX – digital, frequency, and PWM inputs PDD – digital data inputs PAN – angular inputs, engine configuration
Output Drivers	PDX – digital, H-bridge, and PWM outputs PAN – angular outputs, engine configuration PAX – analog/ constant current outputs
Communication Drivers	PCX – CAN IO with and without CAN database (DBC files) PCP – CCP setup and configuration PISO – ISO15765 based diagnostic communication PSMC – UART communication with secondary micro (M560/ M580 ECU specific) PV2G – Vehicle to Grid communication (M560/ M580 ECU specific)

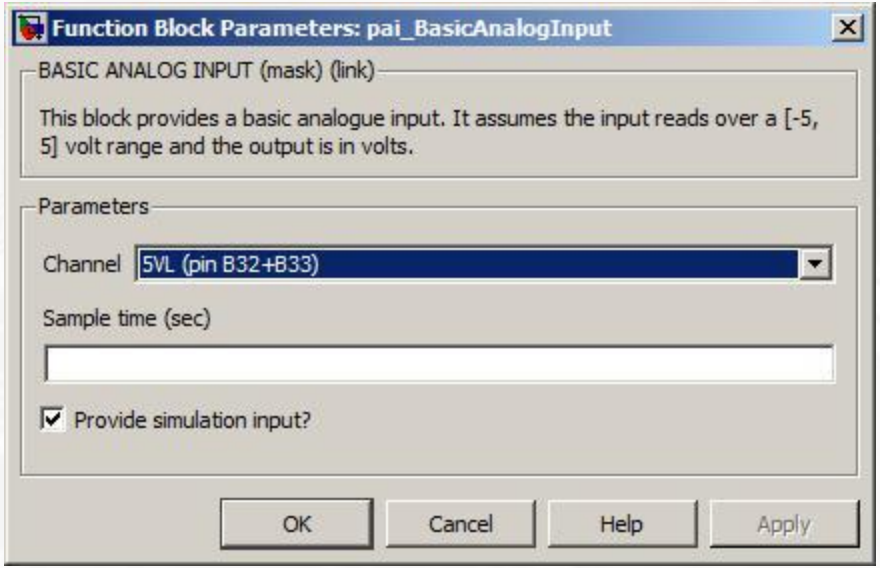
API Layer

Interface type	Functionality
Non-volatile memory	PNV – adaptive calibrations (scalar, maps, arrays), and NV file system access
OBD diagnostic support	PDTC – diagnostic trouble codes PPID – parameter data PFF – diagnostic freeze frames PPR – diagnostic performance ratios PDG – KWP+UDS specific configuration, UDS routine control PJ1939 – SAE J1939 specific configuration and messaging
Misc./System/OS/ Others	PTM – system time PKN – task scheduling PUT – ECU identification and versioning, reset control PREG – embedded registry PCFG – ECU specific configuration PSC – ECU health monitors, versioning, processor loading, stack and reset monitoring

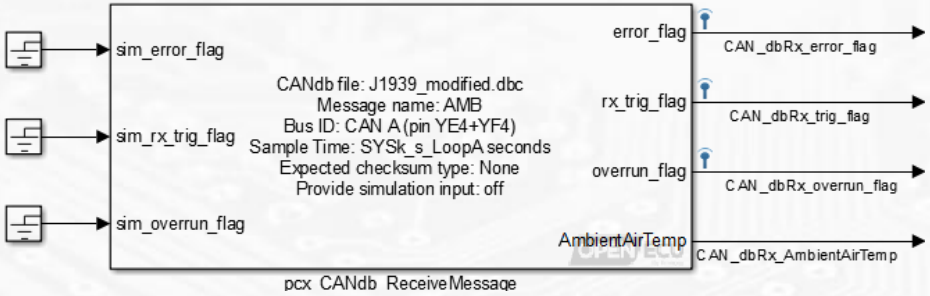
Simulink API example – Analog input

Analog voltage is that seen

at the **micro** pin – not the connector



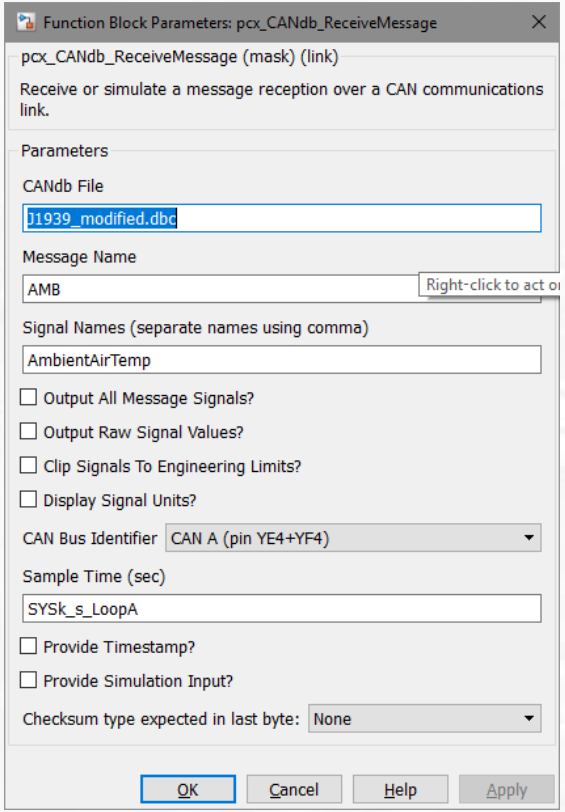
Simulink API example – CAN receive



error_flag in transmit and receive blocks

overrun_flag in receive block detects missed messages – CAN rx blocks only report data of the most recently received message

rx_trig_flag in receive block can be used to detect message time-out with correct application logic



Summary and Outlook

- Embedded control software development has been augmented by MBD and RCP
- Consistent environment for RCP development stage and target ECU implementation stage can be key in saving cost and effort
- OpenECU can provide a complete hardware and software solution to accelerate ECU development and production deployment
- Hardware and software interaction already handled, user focuses on application software
- OpenECU software platform architecture details were discussed, along with API available to the application SW

Thank you!

(end)

OpenECU LLC
47047 W. Five Mile Road
Plymouth
MI 48170-3765
United States of America

+1 734 656 0140

Questions?

openecu.com