# IAR Visual State

*Increase efficiency with graphical modeling*

IAR Visual State is a tool for design and code generation. It is used to graphically design state machines and generate C,C++, C# or Java source code for embedded systems and smartphone/ desktop apps.

## State machine development

When starting to develop an embedded application, there might be several challenges around complexity:

- As complexity increases, how do you ensure that you capture the complete design in code?
- How can you make sure you keep on schedule when you add complexity to your project?
- How can you ensure all parts works together when you are a large team developing a project?

To assist you in solving these challenges, state machines can be used. State machines is a commonly used abstraction that is used when programming state-driven applications, where system behavior is a function of both input and current system state, like user interfaces with complex priorities and animations in HMI (Human-Machine Interface) systems. In addition, state machines are often used as a common language that everybody in a team understands, from beginners to experts.

Using the state machine design solution IAR Visual State enables you to efficiently build and manage large, complex designs. Applications that can benefit from IAR Visual State include automotive applications like instrument clusters, self-driving vehicles systems, advanced power tools, vending machines, HVAC systems, tracking systems, elevators and finally PLC's and control systems.

IAR Visual State is made for embedded systems, enabling you to use state machines in an easy and intuitive way, with no unnecessary features to maneuver among. It is also an excellent tool for design tasks dealing with functional safety as functional safety standards, for example the IEC 61508 standard, recommends state machines as one design method to meet higher SIL levels.

## Reasons to choose IAR Visual State

- Shortening time to market
  - Speeding up time to protoype through smart features
  - Enabling efficient organization and modularization of a design
- Generating very compact code
  - C/C++ code is MISRA C:2012 compliant
  - Java and C# for apps development
  - Architecture agnostic (8,16,32 and 64-bit) and compatible with various RTOS/OS
- Support for hardware debugging with direct feedback in the state machine design
  - Instantly see which state configuration is active and which transition got you there via RealLink if you have another development tool than IAR Embedded Workbench
- Tightly integrated with the IAR Embedded Workbench IDE
  - The generated source code files by IAR Visual State are handled automatically via the project connection file in the IDE
  - The C-SPYLink plugin enables high-level state machine model feedback directly to the IAR C-SPY Debugger including graphical animation in the state machine
- Automatically generates documentation
- Advanced validation and verification capabilities
  - Find issues in your design that are virtually impossible to write full coverage test suites for. Examples of what can be found are dead lock situations, unreachable parts of the design, never consumed input etc.
- Suited for low code development in embedded systems and app's development

## Key features

- Cross-platform Host support
  - o IAR Visual State runs on Linux (Ubuntu) and Windows
- Based on UML (Unified Modeling Language)
- Support for Variant Handling
  - o Model and build different variants from the same Visual State model
- Hierarchical Coder (HCoder) available
  - o Improve the code size and speed efficiency as a complement to classic code generator
- Diff tool
  - o For comparing changes in the state machine in XML, GUI, and file level
- Support for multiple users
  - o Improved support with element files and more
- Support for requirements
  - o The Designer has support for importing requirements from the standard ReqIF format, and the Designer can mark items as fulfilling selectable requirements.
- XMI import/export
  - o Allows to import/export top level state machines to/from standard XMI format, e.g. to exchange models between different tools
- And more:
  - o Images in notes
  - o Copy/paste of state reactions

**IAR Visual State**