## Creating a new project

Launch DAVE IDE and select a directory as the workspace (to save and access the project).

A screenshot of a computer

Description automatically generated

After DAVE IDE is loaded, “File” > “New” > “DAVE Project” to create a new project within the workspace.

A screenshot of a computer

Description automatically generated

Give it a “Project Name” and select an appropriate project type.

A screenshot of a project

Description automatically generated

**Project Types**

* Simple Main Project – equipped with ‘startup files’, a set of low-level device drivers, CMSIS (Common Microcontroller Software Interface Standard) libraries, linker script and empty main function. For user to expand on the project.
* Easy Start Project – Build on an empty project and provides a toy ADC application.
* DAVE CE (Code Engine) Project – Access to full power of CE projects. Includes various drives and startup files, access to configurable applications library (DAVE APPs – xxx).
* Empty Project – Does not contain a main function. Copy completed application developed outside of DAVE into the project and start using the IDE. DAVE will automatically create makefiles and build environment.

Select the corresponding Microcontroller for the Project to ensure the respective libraries are loaded.

A screenshot of a computer program

Description automatically generated

**What is included in the Project file?**

Dave

* Generated – .c and .h files for added DAVE APPs will appear under this folder
* Model – Documentations and other files for added DAVE APPs

Libraries

* CMSIS – enables consistent device support and simple software interfaces
* XMCLib – XMC low-level peripheral drivers for chosen microcontroller
* NewLib – stubs (placeholders) for implementation

Startup

* Containing vector table, default interrupt handlers and program loader

Main.c

* The main application code, implementation of the code (akin to the ‘int main(void)’)

A screenshot of a computer program

Description automatically generated

Under the “APP Dependency Tree”, all APPs used in the project will appear under this section. “Add New APP” to select APPs from the library and add into the Project.

A screenshot of a computer

Description automatically generated

Make sure to “Generate Code” after addition of any APPs to generate the necessary files required for the APP to work.

A screenshot of a computer

Description automatically generated



**Toolbar**

1. Build Active Project – ‘Compile’ the Project
2. Rebuild Active Project – ‘Re-compile’ the Project
3. Add New APP – Opens the DAVE APP Library and select APPs to add into Project
4. Install DAVE APP / Example / Device Library – Install additional APP / Example Project files / Device’s Library from local directory
5. Report – View “Resource Mapping”, “Pin Allocator”, “Signal Assignment” and “APPs” in a table form.
6. Manual Pin Allocator – Manual I/O pin assignment in a table form
7. Manual Resource Assignment – View resource requirements of APPs
8. Global Interrupt – Configure interrupt pre-emption priority and sub-priority
9. Generate Code – Process of generating all necessary code for APPs within Project
10. Pin Mapping Perspective – GUI for the pin assignment of the chipset
11. Debug – Opens up Debug Configuration and begin process of debugging onto microcontroller

Connect the microcontroller via USB and begin to Debug. You will initially see only ‘Debug Configurations…’.

A screenshot of a computer

Description automatically generated

Click the ‘New’ button or double click “GDB SEGGER J-Link Debugger” to create a new debugging configurations. Ensure that J-Link Debugger has been properly installed on the laptop/desktop.

A screenshot of a computer

Description automatically generated

Check that it is debugging the correct project and project file. If the C/C++ Application file is empty, either ‘Build Active Project’ first or check for compilation errors within your code.

A screenshot of a computer program

Description automatically generated

You can amend the ‘timeout’ (in ms) from 0 to a higher number should there be timeout issues when it attempts to start up J-Link Debugger or attempts to setup a connection with the microcontroller. Also ensure that the ‘Device Name’ matches the microcontroller model, else the connection will fail to be set up.

A screenshot of a computer

Description automatically generated

Once debugging process is done, the Debug tab should open. Press the ‘Resume’ or F8 key to start the program.

**APP Dependency**

A dependency tree listing all APPs and their dependencies.

A screenshot of a computer

Description automatically generated

**HW Signal Connectivity**

Visualise the ‘hardware’ signal connections of all the APPs.

A screen shot of a computer

Description automatically generated

Right-click on the individual window to configure the instance of the APP

* Configure APP Instance – opens up the GUI to configure the settings
* Rename Instance Label – Rename the instance to easily call upon in the Project (akin to naming a variable)
* HW Signal Connections – Hardware connection between Source Signal to a different (target) APP Instance
* Manual Pin Allocator – Input/Output pin assignment (eg. DIGITAL\_IO as output to user-configurable LED1 on P0.1)

A screenshot of a computer

Description automatically generated

## C Programming

.c files contains the implementation of the code

.h files provide interface that allow a file to access functions, global variables, macros from other files.

## Micrium uC-Probe XMC

To provide real-time insight into the behavior and performance of embedded systems during development and debugging.

1. **Real-time Monitoring:**
   * uC-Probe allows developers to monitor real-time data and events within the embedded system. This can include variables, memory usage, and other system parameters.
2. **Visualization:**
   * The tool provides visualization capabilities, allowing developers to graphically represent data trends, system states, and other relevant information. This can be helpful for understanding system behavior over time.
3. **Performance Analysis:**
   * Developers can use uC-Probe to analyze the performance of the embedded system. This includes profiling code execution, identifying bottlenecks, and optimizing system performance.
4. **Data Logging:**
   * uC-Probe can log data over time, enabling developers to analyze system behavior during extended periods. This is particularly useful for capturing intermittent issues or observing long-term trends.
5. **Custom Probes and Instruments:**
   * The tool often allows developers to create custom probes and instruments tailored to the specific needs of their application. This can include monitoring specific registers, peripherals, or custom variables.
6. **Integration with IDEs:**
   * uC-Probe is typically integrated with popular Integrated Development Environments (IDEs) and toolchains, making it easier for developers to incorporate monitoring and analysis into their existing development workflow.