

About EWARM

Support Overview

The following EWARM (Embedded Workbench for ARM) compiler/environment versions are supported:

- IAR EWARM 5.3x.
- IAR EWARM 5.4x.
- IAR EWARM 5.5x.
- IAR EWARM 6.1x (C-only).
- IAR EWARM 6.3x (C-only).
- IAR EWARM 6.6x.
- IAR EWARM 7.4x.
- IAR EWARM 7.8x.
- IAR EWARM 8.11x
- IAR EW ARM 8.20.x (Static Analysis Only)

IAR EW integration is provided as follows:

- The preferred method of importing EWARM projects is to use the `cpptesttrace` utility. See [Importing Projects with IAR Embedded Workbench Support](#) for more information.
- C++test also provides a built-in GUI-based EW project importer for EWARM 5.3x-6.3x. The importer is not supported and not recommended for other versions of EWARM.
 - The options extractor works directly with IAR EW project files (`.ewp` files).
 - The project importer accepts both project (`.ewp`) and workspace (`.eww`) files.

The following components are provided to facilitate testing IAR Embedded Workbench projects:

- Compiler configurations for specific supported versions of IAR compiler for ARM (listed above).
- "Use options from IAR Embedded Workbench project" options source.
- "Import IAR Embedded Workbench projects" importer GUI.
- CLI/Batch mode importer: "-ewp" option + "localsettings" properties: "ewp.import.config", "ewp.import.linked", "ewp.import.location", "ewp.import.subdirs", "ewp.import.pathvar.enabled", "ewp.import.pathvar.name", "ewp.import.pathvar.value".
- "IAR_iccarm.mk" Runtime Library build configuration file for building C++test Runtime Library with IAR ARM compiler versions starting from v. 6.1x using 'make'.
- "IAR_iccarm_5_5x_and_older.mk" Runtime Library build configuration file for building C++test Runtime Library with IAR ARM compiler versions 5.3x-5.5x using 'make'.
- Test Configurations prepared for launching Unit & App Testing on C-SPY Simulator:
 - Run IAR EW Tests (Batch Template) - uses EW-generated batch scripts (`.cspy.bat`) to launch simulator.
 - Run IAR EW Application with Mem Monitoring (Batch Template) - uses EW-generated batch scripts (`.cspy.bat`) to launch simulator.
 - Run IAR ARM Tests - a pure manual simulator configuration.
 - Run IAR ARM Application with Mem Monitoring - a pure manual simulator configuration.

Known Limitations

- Some option descriptions may be missing from project files that originate from old EW versions or were generated by other means. If this occurs when the GUI-based EW project importer is used, C++test will assume that the corresponding compiler/linker options are absent and will rely on tool defaults. This may conflict with EW's approach of providing internal defaults for options with missing descriptions.
- The C++test GUI-based EW project importer does not support the following IAR EW project (`.ewp`) settings:
 - **General Options>Library Configuration>CMSIS>DSPLibrary** setting is ignored (regarding linker options).
Workaround: Manually append appropriate library in **Parasoft>C++test>Build Settings>Linkeroptions**
 - The **General Options>Target>FPU** setting is ignored.
Workaround: Manually add appropriate `--fpu` option to **Project Properties >Parasoft>C++test>Build Settings>Compiler Options**.
 - The **Linker>List>Generate linker mapfile** setting is ignored.
Workaround: Manually add the appropriate `--map` option to **Project Properties >Parasoft>C++test>Build Settings>Linker Options**.
 - The `.no_neon-` designated core names are not provided to the `--cpu` option for Cortex-Ax cores, regardless of the **General Options>Target>FPU** setting.
Workaround: Switch **Use options from IAR Embedded Workbench project** mode to **Specify all options manually in Project Properties>Parasoft>C++test>Build Settings** and manually adjust all options.
- Due to the limitation of the preprocessor included with some IAR ARM compiler versions, C++test will not accept the following code for IAR ARM compiler v. 7.2x and earlier:

```
#define MHZ *1000000L
#define FREQ (1MHZ)

void foo(long freq);

void bar(void)
{
    foo(FREQ);
}
```

Workaround: space should be inserted between 1 and MHZ:

```
#define FREQ (1 MHZ)
```

- C++test does not support the `__nounwind` keyword because exceptions are re-thrown after they have been caught. If your program contains `__nounwind`, add the following macro to compiler options under **Project Properties> Parasoft> C++test> Build settings**:

```
-DCPPTTEST_COLLECT_STACK_TRACE=0
```

- C++test does not correctly reconstruct multibyte characters in C.