PUMPKIN™
REAL-TIME SOFTWARE

750 Naples Street • San Francisco, CA 94112 • (415) 584-6360 • http://www.pumpkininc.com

General Instructions for Configuring Salvo Projects

Introduction

Each Salvo distribution is accompanied by Application Notes that explain in detail how to create working Salvo applications with the development tools that Salvo supports for a given target.

This Application Note is intended to aid users in configuring development tools — especially Integrated Development Environments (IDEs) — that are not yet formally certified for use with Salvo.

Before You Begin

This Application Note assumes that Salvo is installed in its default location (`c:\salvo`). Other locations are permitted, but may require changes to the instructions below.

Furthermore, it assumes that you have organized your work on a project basis, either implicitly (in the case where the development tools are project-based) or explicitly (e.g. via a `Makefile`).

You will need to refer to the documentation for your development tools in order to implement the items below. In many cases, IDEs will already have support for these items. If not, you may have to use command-line arguments to the various toolset components (e.g. compiler, linker, etc.).

created by Andrew E. Kalman on Feb 13, 2003 updated on Feb 13, 2003

Required Items

Additional Include Paths

Every file that uses Salvo's API must include the Salvo header file, e.g.:

```
#include "salvo.h"
```

This applies to all of Salvo's source files, as well as any user files that reference the Salvo API. Including the Salvo header file `salvo.h` will cause a variety of additional Salvo header files – as well as the Salvo configuration file – to be included in the source file. *You must take steps to ensure that the compiler can find these header files.* This is usually accomplished either through the IDE or explicitly as command-line arguments to the compiler.

Salvo Header Files

All of Salvo's header files are located in `c:\salvo\inc`. *Therefore every source file* (`*.c`)[1] *must have* `c:\salvo\inc` *specified as an additional include path.*

Tip If your development tools support it, you can usually apply this additional include path to all of your source files. For example, some IDEs allow you to specify additional include paths project-wide.

Salvo Configuration File

We recommend that you place your project-specific Salvo configuration file `salvocfg.h` in the same directory as your own source files (and project files, if using IDE-based tools). Some development tools will automatically search this directory (because it's where the project file resides). Others will not.

You must ensure that the compiler can find your project's `salvocfg.h` file. If you get build errors saying that `salvocfg.h` cannot be found, *then you will have to explicitly add an appropriate additional include path to every source file.*

Salvo Source Files

Occasionally, Salvo Pro users may wish to include Salvo source files in other files instead of linking to Salvo libraries or Salvo

object files. When you do this, *you must specify* `c:\salvo\src` *as an additional include path for any file that includes a Salvo source file*, e.g. via:

```
#include "timer.c"
```

Building on Pre-existing Salvo Projects

Each Salvo distribution comes with ready-to-build tutorial and example projects. Many Salvo beginners will start with these projects and develop them into their own. *To build them successfully with tools that are not yet certified will require a few, non-obvious extra steps.*

Defining the Project's SYS Symbol

Each project in a Salvo distribution is built for a particular development tool and target. In order to build these projects successfully, a compiler- and target-specific symbol must be defined for each preexisting Salvo project source (`*.c`) file. The `SYS` symbol codes (e.g. `SYSF`, for the HI-TECH PICC-18 compiler) are listed in the Salvo User Manual. Their use is limited to compiler- and implementation-specific issues (e.g. the declaration of interrupt service routines) in project source files (e.g. `main.c`, `isr.c`).[2]

If your unsupported tool is an IDE, but the compiler you are using is certified for use with Salvo, then refer to the User Manual and *define the appropriate SYS symbol for every project source file.*

If your unsupported tool is a compiler, then you must review the use of the `SYS` symbol in the project's source files. If your compiler's syntax matches another one exactly, defining the symbol for that compiler may suffice. Otherwise you will have to create additional source code to implement the function(s) that the defined symbols enable in the project's source code.

Tip Some development tools will allow you to simply define a symbol without specifying its value. Others require that you assign a defined symbol a value. When in doubt, assign the defined symbol the value of `1`.

In any case, there is no need to define `SYS` symbols for Salvo source files like `c:\salvo\src\mem.c`.

Symbols for Use with the Project's salvocfg.h

The following symbols are used in Salvo tutorial and example projects to conditionally control compilation of the `salvocfg.h` header file:

`MAKE_WITH_FREE_LIB`	Salvo Lite library build
`MAKE_WITH_TINY_LIB`	Salvo tiny library build
`MAKE_WITH_SE_LIB`	Salvo SE library build
`MAKE_WITH_STD_LIB`	Salvo LE, Pro library build
`MAKE_WITH_SOURCE`	Salvo Pro source-code build

If you are using the project's `salvocfg.h`, *you must define one of these symbols*[3] *for each preexisting Salvo project source (`*.c`) file*, e.g. a project's `main.c`. These symbols do not need to be defined for Salvo source files (e.g. `c:\salvo\src\mem.c`).

If you create your own `salvocfg.h` that does not have these symbols in it, then there is no need to define any of these symbols.

Setting Include Paths

Salvo tutorial and example projects often have their `salvocfg.h` header file and project file in a subdirectory of that where the main project source files (e.g. `main.c`, `isr.c`) reside. *Therefore you may need to add additional include paths to ensure that the compiler can find all of the related files.* Depending on the tools, this may require additional include paths to the main directory, or to the subdirectory, or both.

External Files

Salvo tutorial and example projects sometimes use files that are not located in the project's directory or subdirectory. *If you cannot locate a particular function that is called from one of the project's source files, check for it in the project source files of other, closely related projects.*

For example, some files in `c:salvo\tut\tu1` and its subdirectories are used in tutorials `tu2-tu6`.

Optional Items

Group / Folder Names

We recommend that when creating a new project, you create new Groups / Folders with the following self-explanatory names:

Listings	*for map files, etc.*
Salvo Configuration File	*for* `salvocfg.h`
Salvo Help Files	*for abstracts*
Salvo Libraries	*for Salvo libraries*
Salvo Sources	*for Salvo source files*

and place the corresponding Salvo files within them.

Additional Issues

Absolute / Relative Pathnames

Support for relative pathnames varies among toolsets. Some do not support them at all. Some support them fully with the implicit concept of a "current project directory as home". Others support paths relative to tool-specific environment variables (e.g. `$PROJ_DIR$`, `$TOOLS_DIR$`).

Wherever possible, we recommend that you use relative pathnames (e.g. `..\mydir`) in include paths to enhance portability.

Drive Letters

The need for drive letters (e.g. `c:`, `d:`, etc.) in paths varies among toolsets.

Wherever possible, we recommend that you avoid drive letters (e.g. use `\salvo\inc` instead of `c:\salvo\inc`) in include paths to enhance portability.

Caveats

The PATH Environment Variable

We do not recommend that you add any paths to Salvo's directories to your system's `PATH` environment variable. *Keep the references to Salvo's directories local to your projects.*

Conclusion

Successfully building Salvo projects with as-yet-unsupported tools generally requires just the addition of include paths to the source files of a project. Additional symbols must be defined if the Salvo tutorial and example projects are to be built successfully.

---

[1] It is generally unnecessary to add a Salvo include path to Salvo assembly files.

[2] Please note the distinction between project source files and Salvo source files. Salvo source files are located in c:\salvo\src. Project source files may be part of a Salvo installation, but they are located elsewhere and are not part of the core Salvo source code.

[3] Which one you use depends on the kind of Salvo distribution you have.