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

Using Salvo with HI-TECH PICC & PICC-18 Demo Compilers

Introduction

HI-TECH Software (http://www.htsoft.com/) provides ANSI C compilers for the full range of Microchip (http://www.microchip.com/) PICmicro® MCUs.

Detailed instructions for building Salvo applications for PICmicro® MCUs with the full versions of HI-TECH's compilers can be found in Pumpkin Application Notes AN-1 *Using Salvo Freeware Libraries with the HI-TECH PICC Compiler* and AN-4 *Building a Salvo Application with HI-TECH PICC and Microchip MPLAB*. These Application Notes assume the user has the full version of the PICC or PICC-18 compilers.

HI-TECH also provides time-limited demo versions of their compilers for evaluation. This Application Note lists the additional steps required to successfully build a project when using these demo compilers with Salvo. The Microchip MPLAB IDE is used to illustrate the build environment.

Limitations of Demo Compilers

The limitations of the HI-TECH demo compilers that affect Salvo projects are the lack of support for the `-I` (set include path) and `-D` (define symbol) command-line arguments.

Overview

The demo compilers' lack of certain command-line arguments requires an alternate approach to successfully compile and link Salvo projects. The demo compilers can only find header files located in the same directory as the including source file(s), or in the compiler's include directory. Since only the default include paths are available, you must copy certain files to new locations so

that they are properly included. Additionally, you'll create one more files with the required symbols for inclusion in the project.

Five Steps to Success

Step 1 – Install the Compiler, Open a Project

After installing the demo compiler and Salvo Lite for PICmicro® MCUs,[1] open a Salvo Lite project or follow the instructions in Pumpkin App Notes AN-1 and/or AN-4 to create your own project. Building the project at this point will result in the compiler error:

```
Cannot open include file "salvo.h" (error)
```

This error occurs because the demo compiler's include search path does not include `\salvo\inc`.

Step 2 – Copy Salvo Header Files

Copy all of the files in `\salvo\inc` to the compiler's include directory. The compiler always searches this directory for files to include. For PICC, the default directory is `\ht-pic\include`. For PICC-18, it's `\htsoft\pic18\include`. This will enable the compiler to find the primary header file `salvo.h` as well as the secondary ones. Building the project at this point will result in the compiler error:

```
Cannot open include file "salvocfg.h" (error)
```

This error occurs because the demo compiler's include search path does not include the directory in which the project's configuration header file `salvocfg.h` resides.

Note Step 2 need only be performed once.

Step 3 – Copy Project Configuration File

Copy your project-specific configuration file `salvocfg.h` to the compiler's include directory listed in Step 1. If you are building your own project, then it should successfully compile and link after this step.

Step 4 – Define Symbols

Note Steps 4 and/or 5 are only required when attempting to build a project supplied in a Salvo distribution.

Projects included in Salvo distributions typically require one or more defined symbols for a successful build.[2] Rather than adding these definitions to each source file that requires them, we recommend that you create a header file in the compiler's include directory, define the symbols therein, and include that header file in each of the project's source files that requires the definitions.

Step 5 – Locate Additional Header Files

Multi-part Salvo projects that are spread across multiple directories (e.g. tutorials `tu1-tu6`) often share header files located in the first project's folder.[3] The demo compilers can only find a header file if it is located in the same directory as the source file, or if it is in the compiler's include directory. If a project-specific header file cannot be found, you must copy it from its default location to the same directory as the source file that includes it.

Example 1 – Salvo Demo d4 & PICC

Steps 1-2

Install the HI-TECH PICC demo compiler and copy all the files in `\salvo\inc` to `\ht-pic\include`.

Step 3

Copy the configuration file `\salvo\demo\d4\sysa\salvocfg.h` to `\ht-pic\include\salvocfg.h`.

Step 4

Open the MPLAB project `\salvo\demo\d4\sysa\d4free.pjt`. Choose Project > Edit Project and examine the Project Files:

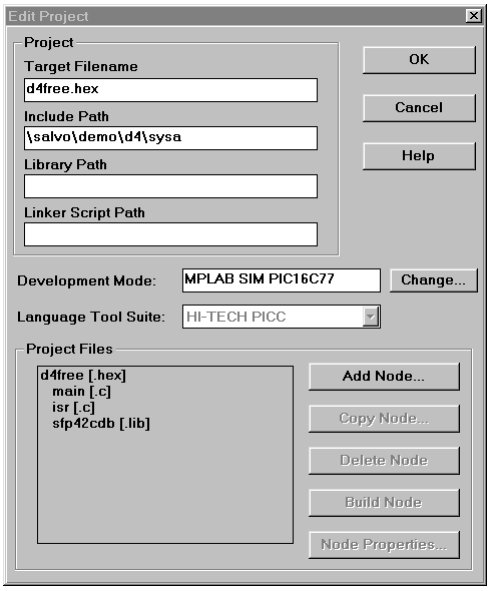

Figure 1: Edit Project Window for demo4

This project has two source file nodes – `main.c` and `isr.c`. `main.c` is located in `\salvo\demo\d4\main.c` and `isr.c` is located in `\salvo\demo\d4\isr.c`.[4] To find the defined symbols for these nodes, select the file and then click on Node Properties:

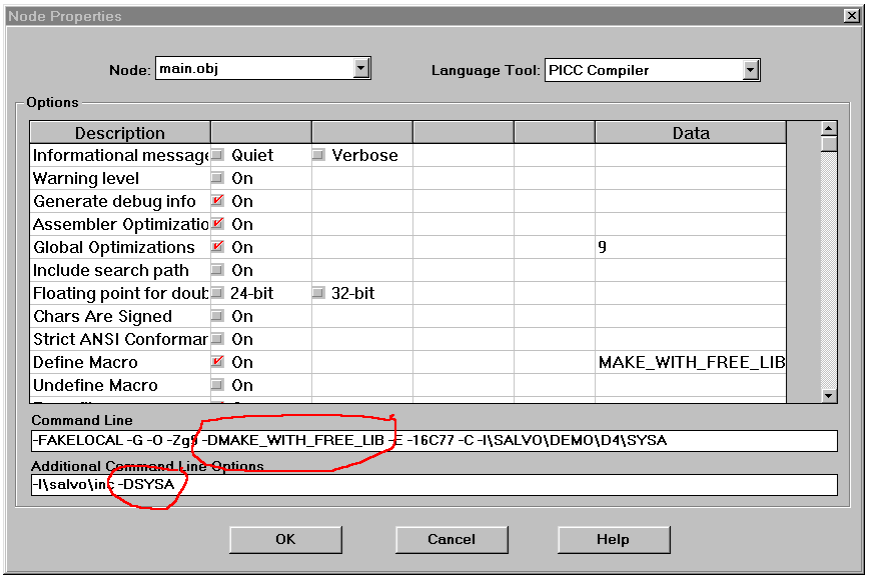

Figure 2: Symbols Defined for Source File Node \salvo\demo\d4\main.c in Project demo\d4

In this project, `main.c` (and `isr.c`) must be compiled with the symbols `SYSA` and `MAKE_WITH_FREE_LIB` defined. Therefore follow *Step 4 – Define Symbols* above by adding:

```
#include "hdrpiccd.h"
```

to the beginning of both `\salvo\demo\d4\main.c` and `\salvo\demo\d4\isr.c`. Create a header file containing:

```
#define SYSA
#define MAKE_WITH_FREE_LIB
```

and save it as `\ht-pic\include\hdrpiccd.h`.[5] The demo project should now build successfully:

```
Building D4FREE.HEX...

Compiling main.c:
Command line: "C:\HT-PIC\BIN\PICC.EXE -FAKELOCAL -G -O -Zg9
-DMAKE_WITH_FREE_LIB -E -16C77 -C -I\SALVO\DEMO\D4\SYSA
-I\salvo\inc -DSYSA \salvo\demo\d4\main.c"
Enter PICC -HELP for help
This compiler will expire in 20 days

Compiling isr.c:
Command line: "C:\HT-PIC\BIN\PICC.EXE -FAKELOCAL -G -O -Zg9
-DMAKE_WITH_FREE_LIB -E -16C77 -C -I\SALVO\DEMO\D4\SYSA
-I\salvo\inc -DSYSA \salvo\demo\d4\isr.c"
Enter PICC -HELP for help
This compiler will expire in 20 days

Linking:
Command line: "C:\HT-PIC\BIN\PICC.EXE -FAKELOCAL -G -Md4free.map
-E -ICD -16C77 -oD4FREE.HEX
\salvo\demo\d4\main.obj \salvo\demo\d4\isr.obj
\salvo\lib\SFP42CDB.lib "
Enter PICC -HELP for help
This compiler will expire in 20 days

Memory Usage Map:

Program ROM   $0000 - $0082  $0083 (   131) words
Program ROM   $058C - $07FF  $0274 (   628) words
                             $02F7 (   759) words total Program ROM
Bank 0 RAM    $0020 - $0036  $0017 (    23) bytes
Bank 0 RAM    $0070 - $0072  $0003 (     3) bytes
                             $001A (    26) bytes total Bank 0 RAM
Bank 1 RAM    $00A0 - $00B6  $0017 (    23) bytes total Bank 1 RAM
Config Data   $2007 - $2007  $0001 (     1) words total Config Data

Program statistics:

Total ROM used      759 words (9.3%)
Total RAM used       46 bytes (12.5%)

Build completed successfully.
```

Figure 3: Build Results with PICC Demo Compiler

Step 5

This step is not required for this project, as all of its source files are contained in the project folder `\salvo\demo\d4`.

Example 2 – Salvo Tutorial tu6 & PICC-18

Steps 1-2

Install the HI-TECH PICC-18 demo compiler and copy all the files in `\salvo\inc` to `\htsoft\pic18\include`.

Step 3

Copy the configuration file `\salvo\tut\tu6\sysf\salvocfg.h` to `\htsoft\pic18\include\salvocfg.h`.

Step 4

Open the MPLAB project `\salvo\tut\tu6\sysf\tu6free.pjt`. Choose Project > Edit Project and examine the Project Files:

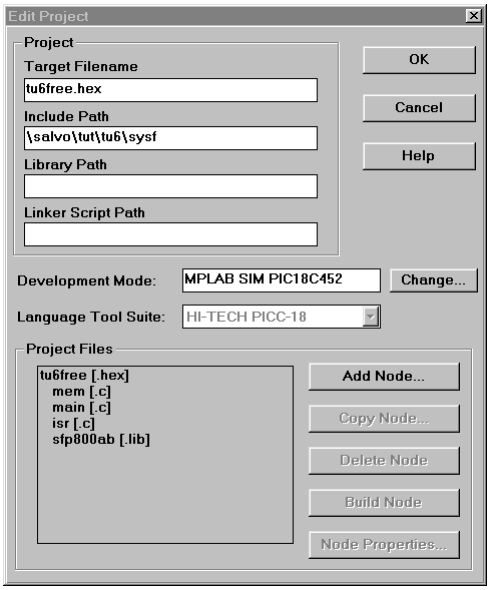

Figure 4: Edit Project Window for demo4

This project has three source file nodes – `mem.c`, `main.c` and `isr.c`. `mem.c` is located in `\salvo\src\mem.c`, `main.c` is located in `\salvo\tut\tu6\main.c` and `isr.c` is located in `\salvo\tut\tu1\isr.c`.[6] To find the defined symbols for these nodes, select the file and then click on Node Properties:

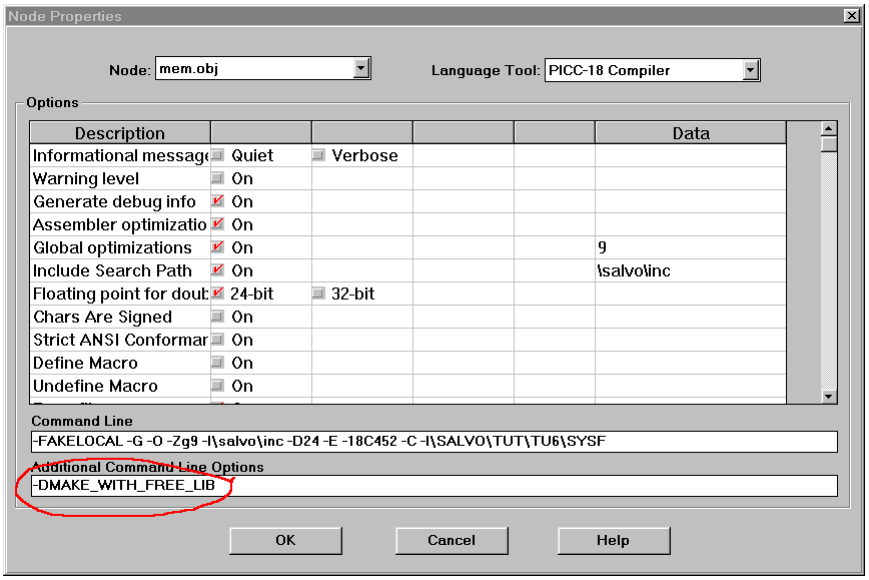

Figure 5: Symbols Defined for Source File Node \salvo\src\mem.c in Project tut\tu6

In this project, `mem.c` must be compiled with the symbol `MAKE_WITH_FREE_LIB` defined.

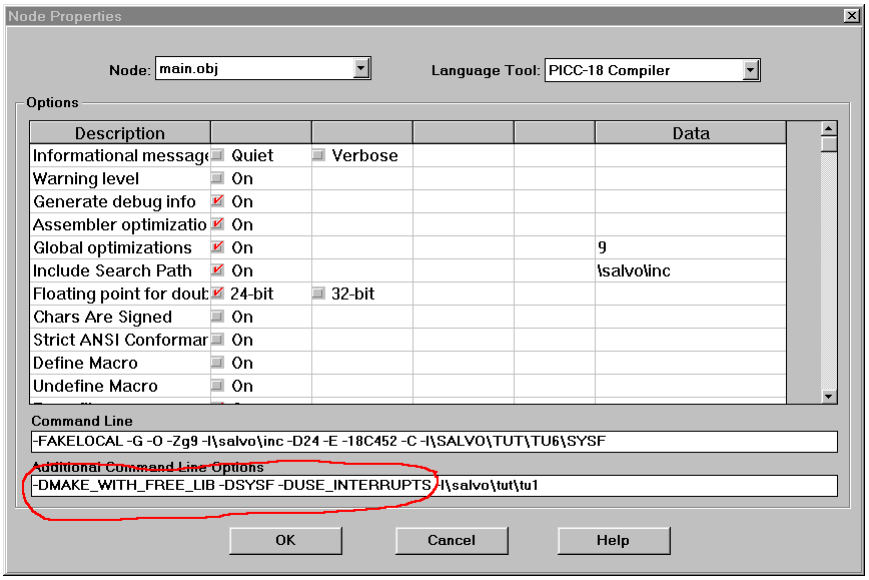

Figure 6: Symbols Defined for Source File Node \salvo\tut\tu6\main.c in Project tut\tu6

In this project, `main.c` (and `isr.c`) must be compiled with the symbols `MAKE_WITH_FREE_LIB`, `SYSF` and `USE_INTERRUPTS` defined. Therefore follow *Step 4 – Define Symbols* above by adding:

```
#include "hdrpiccd.h"
```

to the beginning of `\salvo\src\mem.c`, `\salvo\tut\tu6\main.c` and `\salvo\tut\tu1\isr.c`. Create a header file containing:

```
#define SYSF
#define MAKE_WITH_FREE_LIB
#define USE_INTERRUPTS
```

and save it as `\htsoft\pic18\include\hdrpiccd.h`.

Note In this example, `SYSF` and `USE_INTERRUPTS` are not required for a successful compile of `mem.c`. However, using the same `hdrpiccd.h` header file for all three files simplifies things and is therefore recommended.

Step 5

Attempting to build the project will result in the following error message:

```
Error[ ]   file \salvo\tut\tu6\main.c  14 : Cannot
open include file "main.h"
```

This occurs because the header file `main.h` that is included in this project's `main.c` resides in `\salvo\tut\tu1`, the first of this multi-part tutorial. With the full version of the PICC-18 compiler, this file would automatically be found because of the additional `-I\salvo\tut\tu1` command-line option (see Figure 6).

With the demo version, you must copy `\salvo\tut\tu1\main.h` to `\salvo\tut\tu6\main.h` for a successful compile:

```
Building TU6FREE.HEX...

Compiling mem.c:
Command line: "C:\HTSOFT\PIC18\BIN\PICC18.EXE -FAKELOCAL -G -O -Zg9
-I\salvo\inc -D24 -E -18C452 -C -I\SALVO\TUT\TU6\SYSF
-DMAKE_WITH_FREE_LIB \salvo\src\mem.c"

Compiling main.c:
Command line: "C:\HTSOFT\PIC18\BIN\PICC18.EXE -FAKELOCAL -G -O -Zg9
-I\salvo\inc -D24 -E -18C452 -C -I\SALVO\TUT\TU6\SYSF
-DMAKE_WITH_FREE_LIB -DSYSF -DUSE_INTERRUPTS
-I\salvo\tut\tu1 \salvo\tut\tu6\main.c"

Compiling isr.c:
Command line: "C:\HTSOFT\PIC18\BIN\PICC18.EXE -FAKELOCAL -G -O -Zg9
-I\salvo\inc -D24 -E -18C452 -C -I\SALVO\TUT\TU6\SYSF
-DMAKE_WITH_FREE_LIB -DSYSF -DUSE_INTERRUPTS
-I\salvo\tut\tu1 \salvo\tut\tu1\isr.c"

Linking:
Command line: "C:\HTSOFT\PIC18\BIN\PICC18.EXE -G -FAKELOCAL -INTEL
-Mtu6.map -18C452 -OTU6FREE.HEX
\salvo\src\mem.obj \salvo\tut\tu6\main.obj
\salvo\tut\tu1\isr.obj \salvo\lib\SFP800AB.lib "
Enter PICC18 -HELP for help
This compiler will expire in 20 days

Memory Usage Map:
Program ROM   $000000 - $000003  $000004 (       4) bytes
Program ROM   $000008 - $000013  $00000C (      12) bytes
Program ROM   $000018 - $0009DD  $0009C6 (    2502) bytes
                                 $0009D6 (    2518) bytes total Program ROM
```

```
RAM data      $000010 - $000035  $000026 (      38) bytes
RAM data      $0000F0 - $0000FF  $000010 (      16) bytes
RAM data      $0005E0 - $0005FF  $000020 (      32) bytes
                                 $000056 (      86) bytes total RAM data

Near RAM      $000000 - $00000F  $000010 (      16) bytes total Near RAM
ROM data      $000004 - $000005  $000002 (       2) bytes total ROM data

Program statistics:

Total ROM used    2520 bytes (7.7%)
Total RAM used    102 bytes  (6.6%)   Near RAM used   54 bytes (42.2%)

Build completed successfully.
```

Figure 7: Build Results with PICC-18 Demo Compiler

Note In this project, `isr.c` includes `isr.h`. Since they are both located in `\salvo\tut\tu1`, there is no need to copy `isr.h` to another directory.

Example 3 – Your Own Application

Steps 1-2

Install the HI-TECH PICC or PICC-18 demo compiler and copy all the files in `\salvo\inc` to `\ht-pic\include` (PICC) or `\htsoft\pic18\include` (PICC-18).

Step 3

Create a configuration file `salvocfg.h` for your project and copy it to `\ht-pic\include\salvocfg.h` (PICC) or `\htsoft\pic18\include\salvocfg.h` (PICC-18).

Steps 4-5

These steps are not required for Salvo projects that you create from scratch.

Troubleshooting

A failure to compile and/or link a Salvo project when using a HI-TECH demo compiler is likely due to either a failure to follow the five-step process outlined above, or due to an incorrect `hdrpiccd.h` (or other) header file left over from a previous build. Ensure that the contents of the `hdrpiccd.h` file are correct for the project you are attempting to build. Also ensure that the

`salvocfg.h` header file in the compiler's include directory is the correct one for your project.

When in doubt, uninstall Salvo completely (including deleting the `\salvo` directory and all its subdirectories) and re-install Salvo. This will restore all of the Salvo projects to their correct values.

Precautions

For obvious reasons, *you must undo these steps once you obtain a full version of the HI-TECH compiler*. To do this, uninstall the compiler and Salvo, delete the root directories and all subdirectories of the compiler and Salvo, and then re-install each in its default location.

Note The entire five-step process outlined above *does not require any changes to existing MPLAB projects*.

Conclusion

The HI-TECH demo compilers can be used to build Salvo applications by copying Salvo header and configuration files to the compilers' default include directories.

Projects supplied with Salvo distributions may also require the definition of one or more symbols. It is recommended that these symbols be defined in a user-created header file. In multi-part projects, some header files may need to be copied to new locations.

---

[1] It is assumed that the user is evaluating the demo compiler along with the freeware version of Salvo, Salvo Lite. However, this Application Note applies to all versions of Salvo (Lite, LE, Pro, etc.) for PICmicro® MCUs.

[2] These symbols are used for hardware-specific settings and in the project's `salvocfg.h` configuration header file.

[3] In this case, `salvo\tut\tu1`.

[4] Most source files in a project are located in the project's folder. Shared files may be located outside the project's folder. In MPLAB, use Window > Project to open the Project Window and find the source file's path.

[5] `hdrpiccd.h` for "header file for PICC demo compiler."

[6] See endnote 4.