

JALv2 Compiler Options

JALv2 Compiler Options

There are many options that can be passed to the compiler to tell it, for example, where to find library files, or where to put the output files. These options are all described here.

See the JALv2 documentation for definitions and conventions. Any time multiple options are allowed, the default option is preceded with a '*'. An {empty} option is interpreted as the default option.

All available compiler options can be seen by passing the single options "--help" to the compiler. Use this command to also see the defaults for each option.

Table of Contents

1. File Options	1
1.1. [no-]asm	1
1.2. [no-]codfile	1
1.3. [no-]hex	1
1.4. include	2
1.5. include path	2
1.6. [no-]log	2
1.7. [no-]lst	2
2. Misc.	4
2.1. clear	4
2.2. quiet	4
2.3. task	4
3. Bootloader	5
3.1. bloader	5
3.2. fuse	5
3.3. long start	5
3.4. rickpic	5
3.5. loader18	6
4. Warnings	7
4.1. all	7
4.2. codegen	7
4.3. conversion	7
4.4. directives	7
4.5. misc	7
4.6. range	8
4.7. stack overflow	8
4.8. truncate	8
5. Optimizations	9
5.1. cexpr reduction	9
5.2. const detect	9
5.3. expr reduction	9
5.4. expr simplify	9
5.5. load reduce	9
5.6. temp reduce	10
5.7. variable frame	10
5.8. variable reduce	10
5.9. variable reuse	10
6. Compiler Debugging	12
6.1. codegen	12
6.2. debug	12
6.3. pcode	12
6.4. emu	12
6.5. deadcode	12

Chapter 1. File Options

1.1. [no-]asm

Format:

```
-asm name  
-no-asm
```

Set the name of the generated assembly file to 'name'. The default is the program name with '.jal' replaced by '.asm', or '.asm' appended if the program name doesn't end in '.jal'.

If '-no-asm' is specified, no '.asm' file will be generated.

The assembly file can be compiled by the MPASM and hopefully generate the same HEX file as JALv2.

1.2. [no-]codfile

Format:

```
-codfile name  
-no-codfile
```

Set the name of the generated COD file to 'name'. The default is the program name with '.jal' replaced by '.cod', or '.cod' appended if the program name doesn't end in '.jal'.

If '-no-codfile' is specified, no '.cod' file will be generated.

The COD file is a symbol file used by MPASM and probably other debugging tools. The format was created and is maintained by Byte Craft Limited and its sharing of this format, and general support is very much appreciated.

1.3. [no-]hex

Format:

```
-hex name  
-no-hex
```

Set the name of the generated HEX file to 'name'. The default is the program name with '.jal' replaced by '.hex', or '.hex' appended if the program name doesn't end in '.jal'.

If '-no-hex' is used, no HEX file will be generated.

The HEX file is used by PIC programmers and bootloaders to load the program onto the microcontroller.

1.4. include

Format:

```
-include filename [ ';' filename2... ]
```

include 'filename' before parsing the file. Multiple files can be included when separated by ';' or when multiple '-include' directives are used.

1.5. include path

Format:

```
-s path [ ';' path1... ]
```

Set the include path, elements separated with ';'. Multiple "-s" options append more path elements.

1.6. [no-]log

Format:

```
-log filename  
-no-log
```

Generate a log file which will contain all messages emitted by the compiler. If absent, standard output is used.

1.7. [no]-lst

Format:

```
-lst filename  
-no-lst
```

Set the name of the listing file to filename, or prevent the generation of a listing file. The default is no listing file. The listing file has never been correctly generated, so this option is useless.

Chapter 2. Misc.

2.1. clear

Format:

```
-[no-]clear
```

Clear data area on program entry. This does not clear user placed or variables, or variables marked VOLATILE.

2.2. quiet

Format:

```
-[no-]quiet
```

Turns off the progress messages

2.3. task

Format:

```
-task cexpr
```

Sets the maximum number of concurrent tasks to *cexpr*. The default is 0. Note this value must be one more than the number of concurrent tasks as the main program counts as a task.

nb: It is better to use 'PRAGMA TASK' in your program than setting the value here. Doing so guarantees the value is correct even if you forget to pass it during compilation.

Chapter 3. Bootloader

Bootloaders are tiny programs that allow a chip to be reprogrammed over the serial or USB ports, eliminating the need for extra programming hardware. There are several variants, and each has slightly different requirements of the program it hosts. These are the ones currently defined.

3.1. bloader

Format:

```
-bloader
```

Using the screamer/bloader PIC loader. See "PRAGMA BOOTLOADER BLOADER".

3.2. fuse

Format:

```
-[no-]fuse
```

Put the '___config' line in the assembly or HEX files

3.3. long start

Format:

```
-long-start
```

Force the first generated instruction to be a long jump. See "PRAGMA BOOTLOADER LONG_START".

3.4. rickpic

Format:

```
-rickpic
```

Assumes the target PIC is using Rick Farmer's PIC bootloader. See "PRAGMA BOOTLOADER RICKPIC".

3.5. loader18

Format:

```
-loader18 [ cexpr ]
```

See "PRAGMA BOOTLOADER LOADER18"

Chapter 4. Warnings

4.1. all

Format:

```
-W[no-]all
```

enable/disable all warnings

4.2. codegen

Format:

```
-W[no-]codegen
```

enable/disable code generation warnings

4.3. conversion

Format:

```
-W[no-]conversion
```

enable/disable signed/unsigned conversion warning

4.4. directives

Format:

```
-W[no-]directives
```

enable/disable warning when a compiler directive is found

4.5. misc

Format:

```
-W[no-]misc
```

enable/disable uncategorized warnings

4.6. range

Format:

```
-W[no-]range
```

enable/disable value out of range warnings

4.7. stack overflow

Format:

```
-W[no-]stack-overflow
```

issue a warning on hardware stack overflow instead of an error

4.8. truncate

Format:

```
-W[no-]truncate
```

enable/disable possible truncation in assignment warning

Chapter 5. Optimizations

5.1. cexpr reduction

Format:

`-[no-]cexpr-reduction`

enable/disable constant expression reduction

5.2. const detect

Format:

`-[no-]const-detect`

enable/disable constant detection

5.3. expr reduction

Format:

`-[no-]expr-reduction`

enable/disable expression reduction

5.4. expr simplify

Format:

`-[no-]expr-simplify`

enable/disable expression simplification

5.5. load reduce

Format:

`-[no-]load-reduce`

enable/disable redundant load of W removal

5.6. temp reduce

Format:

`-[no-]temp-reduce`

enable/disable temporary reduction

5.7. variable frame

Format:

`-[no-]variable-frame`

allocate variables into a full frame

5.8. variable reduce

Format:

`-[no-]variable-reduce`

enable/disable unused or unassigned variables removal

5.9. variable reuse

Format:

`-[no-]variable-reuse`

enable/disable reusing variable space

Chapter 6. Compiler Debugging

These options are most useful for debugging the compiler itself.

6.1. codegen

Format:

`-[no-]codegen`

do not generate any assembly code

6.2. debug

Format:

`-[no-]debug`

show debug information

6.3. pcode

Format:

`-[no-]pcode`

show pcode in the asm file

6.4. emu

Format:

`-[no-]emu`

Run the emulator after compiling.

6.5. deadcode

Format:

`-[no-]deadcode`

enable dead code elimination