JALv2 Compiler Options

Contents

Chapter

1

Introduction

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.

Chapter

2

File Options

[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.

[no-]codfile

Format:

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

Set the name of the generated COD file to `name'. The defaultis 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.

[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.

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.

include path

Format:

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

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

[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.

[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

3

Miscellaneous

clear

Format:

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

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

fastmath

Format:

```
-[no-]fastmath
```

Use the fastmath versions for multiply.

Without this flag, only one multiply routine is created which can handle the largest types ever multiplied. This creates smaller programs, at the expense of runtime speed.

When this flag is set, a different multiply routine is generated for each types multiplied. This results in fast code, at the expense of program size. A side effect of this is that any multiplication routine that is only used once is inlined.

quiet

Format:

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

Turns off the progress messages.

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

4

Bootloader

Booloaders 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".

fuse

Format:

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

Put the `__config' line in the assembly or HEX files.

long start

Format:

```
-long-start
```

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

rickpic

Format:

```
-rickpic
```

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

loader18

Format:

```
        -loader18 [ cexpr ]
```

See "PRAGMA BOOTLOADER LOADER18".

Chapter

5

Warnings

all

Format:

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

enable/disable all warnings

codegen

Format:

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

enable/disable code generation warnings

conversion

Format:

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

enable/disable signed/unsigned conversion warning

directives

Format:

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

enable/disable warning when a compiler directive is found

misc

Format:

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

enable/disable uncategorized warnings

range

Format:

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

enable/disable value out of range warnings

stack overflow

Format:

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

issue a warning on hardware stack overflow instead of an error

truncate

Format:

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

enable/disable possible truncation in assignment warning

Chapter

6

Optimizations

Optimizations that are enabled by default are considered safe -- they have been well tested and shown to not cause any problems. Optimizations that are disabled by default have not been as well tested. Using them might be unsafe. Before reporting any problems with the compiler, please make sure to retest with only the default optimizations and let me know the outcome.

cexpr reduction

Format:

```
-[no-]cexpr-reduction
```

enable/disable constant expression reduction

Default: enabled

Purpose: Detect expressions or subexpressions composed completely of constants, and reduce these to a single value.

const detect

Format:

```
-[no-]const-detect
```

enable/disable constant detection

Default: disabled

Purpose: Look for variables that are only assigned a single, constant value and replace them with that constant value.

expr reduction

Format:

```
-[no-]expr-reduction
```

enable/disable expression reduction

Default: enabled

Purpose: eliminate or refactor many simple operations and identities:

- x +/- 0 --> x
- x - 0 --> x
- 0 - x --> -x
- x - x --> 0

- x * 0 --> 0
- x * 1 --> x
- x * (-1) --> -x
- x / 1 --> x
- 0 / x --> 0
- x / (-1) --> -x
- x % 0 --> 0
- x % 1 --> 0
- x < 0, x unsigned --> 0
- x <= 0, x unsigned --> x == 0
- x >=0, x unsigned --> 1
- x > 0, x unsigned --> x != 0
- x & 0 --> 0
- x & x --> x
- x | 0 --> x
- x | x --> x
- x ^ 0 --> x
- x ^ x --> 0
- -x, x is single bit --> x
- 0 << x --> 0
- x << 0 --> x
- x << C, where C is >= bit size of x --> x

expr simplify

Format:

```
-[no-]expr-simplify
```

enable/disable expression simplification

Default: disabled

Purpose: Combine common subexpressions.

load reduce

Format:

```
-[no-]load-reduce
```

enable/disable redundant load of W removal

Default: disabled

Purpose: Look for redundant loads into the working register (W) and remove them.

temp reduce

Format:

```
-[no-]temp-reduce
```

enable/disable temporary reduction

Default: disabled

Purpose: Look for assignments to a temporary which is then immediately assigned to another variable and never again used, and remove it.

variable frame

Format:

```
        -[no-]variable-frame
```

allocate variables into a full frame

Default: disabled

Purpose: Place all variables in a function into a single block instead of using the first available space. This can minimize the number of bank switching operations, but can also lead to less efficient use of the data areas.

variable reduce

Format:

```
        -[no-]variable-reduce
```

enable/disable unused or unassigned variables removal

Default: enabled

Purpose: If a variable is assigned a value but never used in an expression, it is removed. Likewise if a variable is never assigned a value, but is used in an expression it is replaced with the constant 0.

variable reuse

Format:

```
        -[no-]variable-reuse
```

enable/disable reusing variable space

Default: enabled

Purpose: If two variables, say X and Y, are never, `live,' at the same time they can share the same data location. This leads to far more efficient use of the data area, but currently can lead to extreme compile times (on the order of hours).

Chapter

7

Compiler Debugging

These options are most useful for debugging the compiler itself.

codegen

Format:

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

do not generate any assembly code

debug

Format:

```
-[no-]debug
```

show debug information

pcode

Format:

```
-[no-]pcode
```

show pcode in the asm file

emu

Format:

```
-[no-]emu
```

Run the emulator after compiling.

deadcode

Format:

```
-[no-]deadcode
```

enable dead code elimination