

## Target Description File

Target description files are text files that contain a series of commands to define how the code generator will output code for a particular target. The general format of a command is as follows:

```
COMMAND=<ARGUMENT>
```

The name of the command is specified first, followed by an equal sign, then the argument surrounded by less-than and greater-than characters is specified. For example, the following command defines the target name.

```
TARGET_NAME=<HC08>
```

Arguments can also extend over multiple lines, and have replaceable parameters. Parameters are special keywords surrounded by a left brace and right brace character. For example, the following command specifies how to write a 16-bit word value to the FPU. The {byte} parameter is replaced by an 8-bit value when the code is generated.

```
WRITE_WORD=<          lda    {byte}
                jsr    fpu_write
                lda    {byte}+1
                jsr    fpu_write>
```

## Commands

In order to be recognized as a target description file, the first line of the file must contain the TARGET\_NAME command. A target description file only needs to contain those commands that are necessary to define the output for a particular target. There are default values for many of the commands. The available commands are as follows:

TARGET_NAME	WRITE_BYTE
MAX_LENGTH	WRITE_WORD
MAX_WRITE	WRITE_LONG
TAB_LENGTH	WRITE_STRING
DECIMAL_FORMAT	
HEX_FORMAT	READ_DELAY
STRING_HEX_FORMAT	READ_BYTE
OPCODE_PREFIX	READ_WORD
COMMENT_PREFIX	READ_LONG
SOURCE_PREFIX	
SEPARATOR	REGISTER_DEFINITION
CONTINUATION	BYTE_DEFINITION
	WORD_DEFINITION
START_WRITE_TRANSFER	LONG_DEFINITION
START_READ_TRANSFER	FLOAT_DEFINITION
STOP_TRANSFER	
WAIT	PRINT_FLOAT
	PRINT_LONG
WRITE	PRINT_FPUSTRING
WRITE_BYTE_FORMAT	PRINT_NEWLINE
WRITE_WORD_FORMAT	PRINT_STRING
WRITE_LONG_FORMAT	
WRITE_STRING_FORMAT	RESERVED_PREFIX
	RESERVED_WORD

## **Tab Spacing**

The horizontal tab (0x09) character, {t} and {tn} parameters are used to align the output to particular character positions, and can be inserted into any of the output commands. The horizontal tab (0x09) character and {t} parameter will insert spaces until the next character position is a multiple of the value specified by the `TAB_SPACING` command. The {tn} parameter will insert characters until the character position equals the value specified. For example, {t17} would insert space up until character position 17.



## **FLOAT\_DEFINITION**

### **Define float variable definition**

FLOAT\_DEFINITION=<*string*>

Default: empty string

Parameters: {name}

Example: FLOAT\_DEFINITION=<float {name};>

Description: This command defines the instruction sequence used to define an 32-bit floating point variable. A carriage return and linefeed character is appended to the end of the output.

---

## **HEX\_FORMAT**

### **Set the prefix for hexadecimal numbers**

HEX\_FORMAT=<*string*>

Default: \$ (dollar sign)

Parameters: {byte}

Example: HEX\_FORMAT=<0x{byte}>

Description: This command sets the prefix character for hexadecimal numbers.

---

## **LONG\_DEFINITION**

### **Define long variable definition**

LONG\_DEFINITION=<*string*>

Default: empty string

Parameters: none

Example: LONG\_DEFINITION=<long {name};>

Description: This command defines the instruction sequence used to define a 32-bit integer variable. A carriage return and linefeed character is appended to the end of the output.

---

## **MAX\_LENGTH**

### **Set maximum length of write instruction**

MAX\_LENGTH=<*length*>

Default: 80

Parameters: none

Example: MAX\_LENGTH=<90>

Description: This command defines the maximum length of a source line.

---

## **MAX\_WRITE**

### **Set maximum number of bytes in write instruction**

MAX\_WRITE=<*n*>

Default: 1

Parameters: none

Example: MAX\_WRITE=<8>

Description: This command defines the maximum number of bytes in a write command.

---

**OPCODE\_PREFIX****Set the prefix for opcodes in WRITE command**

OPCODE\_PREFIX=<*string*>

Default: empty string

Parameters: none

Example: OPCODE\_PREFIX=<FPU\_>

Description: This command sets the prefix for opcodes used in write\_command. It can be used in conjunction with a symbol definition file to ensure unique names for the opcode constants.

---

**PRINT\_FLOAT****Define instructions to print float value**

PRINT\_FLOAT=<*string*>

Default: empty string

Parameters: {byte}

Example: PRINT\_FLOAT=<format = {byte}  
GOSUB PRINT\_FLOAT>

Description: This command defines the instruction sequence to print a 32-bit floating point value. A carriage return and linefeed character is appended to the end of the output.

---

**PRINT\_FPUSTRING****Define instructions to print FPU string**

PRINT\_FPUSTRING=<*string*>

Default: empty string

Parameters: none

Example: PRINT\_FPUSTRING=<GOSUB PRINT\_FPUSTRING>

Description: This command defines the instruction sequence to print FPU string. A carriage return and linefeed character is appended to the end of the output.

---

**PRINT\_LONG****Define instructions to print long value**

PRINT\_LONG=<*string*>

Default: empty string

Parameters: {byte}

Example: PRINT\_FLOAT=<format = {byte}  
GOSUB PRINT\_LONG>

Description: This command defines the instruction sequence to print a 32-bit integer value. A carriage return and linefeed character is appended to the end of the output.

---

## **PRINT\_NEWLINE**

### **Define instructions to print new line**

PRINT\_NEWLINE=<*string*>

Default: empty string

Parameters: none

Example: PRINT\_NEWLINE=<DEBUG CR>

Description: This command defines the instruction sequence to print a new line. A carriage return and linefeed character is appended to the end of the output.

---

## **PRINT\_STRING**

### **Define instructions to print text string**

PRINT\_STRING=<*string*>

Default: empty string

Parameters: {*string*}

Example: PRINT\_STRING=<DEBUG "{*string*}">

Description: This command defines the instruction sequence to print text string. A carriage return and linefeed character is appended to the end of the output.

---

## **READ\_BYTE**

### **Define instructions to read 8-bit value**

READ\_BYTE=<*string*>

Default: empty string

Parameters: none

Example: READ\_BYTE=<{*name*} = fpu\_readByte();>

Description: This command defines the instruction sequence to use to read an 8-bit value. A carriage return and linefeed character is appended to the end of the output.

---

## **READ\_DELAY**

### **Define instructions for read delay**

READ\_DELAY=<*string*>

Default: empty string

Parameters: none

Example: READ\_DELAY=<call fpu\_readDelay();>

Description: This command defines the instruction sequence to be used to wait for the read delay. A carriage return and linefeed character is appended to the end of the output.

---

## **READ\_LONG**

### **Defines command to read 32-bit value**

READ\_LONG=<*string*>

Default: empty string

Parameters: none

Example: READ\_LONG=<{*name*} = fpu\_readLong();>

Description: This command defines the instruction sequence to use to read a 32-bit value. A carriage return and linefeed character is appended to the end of the output.

---

## **READ\_WORD** **Defines instructions to read 16-bit value**

READ\_WORD=<*string*>

Default: empty string

Parameters: none

Example: READ\_WORD=<{name} = fpu\_readWord();>

Description: This command defines the instruction sequence to use to read a 16-bit value. A carriage return and linefeed character is appended to the end of the output.

---

## **REGISTER\_DEFINITION** **Define register definition**

REGISTER\_DEFINITION=<*string*>

Default: empty string

Parameters: {name}, {register}

Example: REGISTER\_DEFINITION=<#define {name} {register}>

Description: This command defines the instruction sequence used to define a register constant. A carriage return and linefeed character is appended to the end of the output.

---

## **RESERVED\_PREFIX** **Define prefix for reserved words**

RESERVED\_PREFIX=<*string*>

Default: F\_ (F and underscore)

Parameters: none

Example: RESERVED\_PREFIX=<FPU\_>

Description: This command defines the prefix to add to reserved words in order to make them unique.

---

## **RESERVED\_WORD** **Define reserved word**

RESERVED\_WORD=<*string*>

Default: empty string

Parameters: none

Example: RESERVED\_WORD=<SIN>

Description: This command defines a reserved word. Multiple RESERVED\_WORD commands can be used, with each command specifying one reserved word.

---

**SEPARATOR****Define separator character for WRITE command**

SEPARATOR=<*string*>

Default: , (comma and space)

Parameters: none

Example: SEPARATOR=<, >

Description: This command sets the separator character used between items in write\_command.

---

**SOURCE\_PREFIX****Set indent for the start of a comment line**

SOURCE\_PREFIX=<*string*>

Default: ; (semi-colon)

Parameters: none

Example: SOURCE\_PREFIX=< ;-->

Description: This command sets the prefix that's added to source code lines that are copied as comments included with the generated code. The correct string must be specified for a valid comment.

---

**START\_READ\_TRANSFER****Define instructions for start of a read transfer**

START\_READ=<*string*>

Default: empty string

Parameters: none

Example: START\_READ=<CALL START\_READ();>

Description: This command defines the instruction sequence used to start a read transfer. Some implementations will not require this command. A carriage return and linefeed character is appended to the end of the output.

---

**START\_WRITE\_TRANSFER****Define instructions for start of a write transfer**

START\_WRITE=<*string*>

Default: empty string

Parameters: none

Example: START\_WRITE=<CALL START\_WRITE();>

Description: This command defines the instruction sequence used to start a write transfer. Some implementations will not require this command. A carriage return and linefeed character is appended to the end of the output.

---

**STOP\_TRANSFER****Define instructions for end of read or write transfer**

STOP=<*string*>

Default: empty string

Parameters: none

Example:       STOP=<CALL STOP ();>

Description:    This command defines the instruction sequence used to end a read or write transfer. Some implementations will not require this command. A carriage return and linefeed character is appended to the end of the output.

---

## **STRING\_HEX\_FORMAT                    Define format for non-printable string characters**

STRING\_HEX\_FORMAT=<*string*>

Default:       empty string

Parameters:    none

Example:       STRING\_HEX\_FORMAT=<\{byte}>

Description:    This command defines the syntax for writing a non-printable character using write\_command.

---

## **TAB\_SPACING                            Set number of characters per tab**

TAB\_SPACING=<*n*>

Default:       4

Parameters:    none

Example:       TAB\_LENGTH=<8>

Description:    This command sets the number of characters in a tab. The absolute value of *n* specifies the number of characters. If *n* is positive, only spaces are used to move to the next tab position. If *n* is negative, then horizontal tabs (0x09) and spaces are used to move to the next tab position.

---

## **TARGET\_NAME                           Define the target name**

TARGET\_NAME=<*target name*>

Default:       none

Parameters:    none

Example:       TARGET\_NAME=<C compiler>

Description:    This command must be on the first line of the file in order for the file to be recognized as a target description file. It defines the name that will appear in the target menu.

---

## **WAIT                                   Define instructions to wait for ready status**

WAIT=<*string*>

Default:       empty string

Parameters:    none

Example:       WAIT=<call fpu\_wait();>

Description:    This command defines the instruction sequence used to wait for the FPU ready status. A carriage return and linefeed character is appended to the end of the output.

---

## **WORD\_DEFINITION**

### **Define word variable definition**

WORD\_DEFINITION=<*string*>

Default: empty string

Parameters: {name}

Example: WORD\_DEFINITION=<int {name};>

Description: This command defines the instruction sequence used to define a 16-bit integer variable. A carriage return and linefeed character is appended to the end of the output.

---

## **WRITE**

### **Define instructions to write bytes**

WRITE=<*string*>

Default: empty string

Parameters: {byte}

Example: WRITE=<call fpu\_write({byte});>

Description: This command defines the instruction sequence used to write bytes to the FPU, and is required for all implementations. A carriage return and linefeed character is appended to the end of the output.

---

## **WRITE\_BYTE**

### **Define instructions to write 8-bit value**

WRITE\_BYTE=<*string*>

Default: empty string

Parameters: none

Example: WRITE\_BYTE=<call fpu\_write({byte});>

Description: This command defines the instruction sequence used to output an 8-bit value. A carriage return and linefeed character is appended to the end of the output.

---

## **WRITE\_BYTE\_FORMAT**

### **Define 8-bit value format for WRITE command**

WRITE\_BYTE\_FORMAT=<*string*>

Default: empty string

Parameters: {byte}

Example: WRITE\_BYTE\_FORMAT=<{byte}>

Description: This command defines the syntax for writing an 8-bit value using the WRITE command.

---

## **WRITE\_LONG**

### **Define instructions to write 32-bit value**

WRITE\_LONG=<*string*>

Default: empty string

Parameters: none

Example: WRITE\_LONG=<call fpu\_writelong({long});>

Description: This command defines the instruction sequence used to output a 32-bit value. A carriage return and linefeed character is appended to the end of the output.

---

### **WRITE\_LONG\_FORMAT**                      **Define 32-bit value format for WRITE command**

WRITE\_LONG=<*string*>

Default: empty string

Parameters: none

Examples: WRITE\_LONG=<{byte}<<24, {byte}<<16, {byte}<<8, {byte}>  
WRITE\_LONG=<{word}(1), {word}(2)>  
WRITE\_LONG=<{long}>

Description: This command defines the syntax for writing a 32-bit value using the WRITE command.

---

### **WRITE\_WORD**                              **Define instructions to write 16-bit value**

WRITE\_WORD=<*string*>

Default: empty string

Parameters: none

Example: WRITE\_WORD=<call fpu\_writeWord{word};>

Description: This command defines the instruction sequence used to output a 16-bit value. A carriage return and linefeed character is appended to the end of the output.

---

### **WRITE\_WORD\_FORMAT**                      **Define 16-bit value format for WRITE command**

WRITE\_WORD=<*string*>

Default: empty string

Parameters: {byte}, {word}

Examples: WRITE\_WORD=<{word}\16>  
WRITE\_WORD=<{byte}<<8, {byte}>

Description: This command defines the syntax for writing a 16-bit value using the WRITE command.

---

### **WRITE\_STRING**                              **Define instructions to write string value**

WRITE\_STRING=<*string*>

Default: empty string

Parameters: none

Example: WRITE\_STRING=<call fpu\_writeString("{string}");>

Description: This command defines the instruction sequence used to output a zero-terminated string value. A carriage return and linefeed character is appended to the end of the output.

---

**WRITE\_STRING\_FORMAT****Define write string format for WRITE command**

WRITE\_STRING=<*string*>

Default: empty string

Parameters: none

Example: WRITE\_STRING=<"{string}">

Description: This command defines the syntax for writing a zero-terminated string using the WRITE command.

---