Migration guide

Migrating from Renesas toolchain for M16C/R8C to IAR Embedded Workbench® for Renesas RL78

Use this guide as a guideline when converting source code written for the Renesas toolchain for M16C/R8C to IAR Embedded Workbench® for Renesas RL78. Locate a feature in the left-hand column; then the IAR-specific counterpart can be found to the right. For detailed information about this feature specific to IAR Embedded Workbench®, see the relevant documentation. For a complete list of guides, see IAR Information Center in the IDE.

	Product	Version number
Migrating from	Renesas C/C++ Compiler Package for M16C Series and R8C Family [M3T-NC30WA]	5.x and 6.x
Migrating to	IAR Embedded Workbench® for Renesas RL78	2.x

Compiler-specific details

Renesas	IAR Systems
Programming languages	
Assembler, C, C++, EC++ (V.6.00 only)	Supported programming languages: assembler, C, Embedded C++, Extended Embedded C++, and C++.
C89	For C, the C99 standard is default, but C89 can optionally be used. C99 is supported by the library.
Processor configuration	
`-R8C`: Generate code for R8C <= 48KB `-R8CE`: Generate code for R8C > 48KB None: Generate code for M16C	`--core={s1|s2|s3}` `s1` Generates code for S1, the RL78 core with only one register bank and a multiplexed 8-bit bus. `s2` Generates code for S2, the core without instructions to support a hardware multiplier/divider. `s3` (default) Generates code for S3, the core with instructions to support a hardware multiplier/divider.
Memory models/Data models/Code models	
Default: near RAM and far ROM Options for near ROM and far RAM	Supported code models (option `--code_model`): near (default): Function calls reach the first 64 Kbytes of memory. far: Function calls reach the entire 1 Mbyte memory.
Default: near RAM and far ROM Options for near ROM and far RAM	Supported data models (option `--data_model`): near (default): Data is by default placed in the highest 64 Kbytes of memory far: Data is by default placed in the entire 1 Mbyte of memory
One library for M16C and one for R8C	The linker automatically selects appropriate libraries.
Overriding default placement of given code/data model	

near, far	To override default placement of the selected code model, use any of these memory attributes:
	`__callt`
	`__near_func` (default)
	`__far_func`

near, far	To override default placement of the selected data model, use any of these memory attributes: `__near` (default) : The highest 64 Kbytes `__far` : The entire 1 Mbyte of memory. Maximum object size 65535 bytes. An object cannot cross a 64-Kbyte boundary. `__huge` : The entire 1 Mbyte of memory. Maximum object size 1 Mbyte. No data model. For example:

	`__near int i = 3;` `__far unsigned u;`

Absolute placement of variables	

`#pragma address variable_name address_value;`
For example:
`#pragma address pm0 0004h`
	`__no_init char a @0x80;`
	
	or
	
	`#pragma location=0x80`
	`__no_init const int a;`

Absolute placement of functions	

`#pragma section program my_section`
`void foo(void)`
`{}`
`#pragma section program program`
	`void f(void) @ "MyFunctions";`
	or
	`void f(void) @ "MyFunctions"`
	`{`
	`}`
	or
	`#pragma location="MyFunctions"`
	`void f(void);`

Place `my_section` at desired address using linker options or `sect30.inc` (V.5.45 only) Place `my_section` at desired address using linker options (V.6.00 only)	The section `MyFunctions` must be placed by customizing the linker configuration file. See the compiler guide section *Customizing the linker configuration file*.
Constants in ROM	
`const unsigned short constants[] = {0x1234, 0x5678}`	`const unsigned short constants[] = {0x1234, 0x5678}`
Interrupt functions	

`#pragma interrupt MyInterruptRoutine (vect=23)`
`void MyInterruptRoutine(void)`
`{}`
	`#pragma vector = 0x17`
	`__interrupt void MyInterruptRoutine(void)`
	`{`
	`  /* Do something here.*/`
	`}`
	or
	`#pragma vector = UART1_R_RXNE_vector /* Symbol from I/O header file */`
	`__interrupt void MyInterruptRoutine(void)`
	`{`
	`  /* Do something here. */`
	`}`
	Note that an interrupt function must have the return type `void`, and it cannot specify any parameters.

Inline assembler	

`asm("mov.b #11h, r0");`
`asm( " MOV.W R0, $$[FB]",f );`
`asm( " MOV.W R0, $$",s );`
`asm( " MOV.W R0, $@",f );`
	`asm["movw ax, sp"];`
	`asm["mov a, 0xff"];`

Renesas		IAR Systems
Sizes on integers and floating-point		
8 bits	`char`	8 bits
16 bits	`int`	16 bits
16 bits	`short`	16 bits
32 bits	`float`	32 bits
32 bits	`long`	32 bits
64 bits	`long long`	32 bits
64 bits	`double`	32 bits (treated as `float`)

Extended keywords	
`asm, _asm`	`asm, __asm`
`_Bool`	`bool` (language extensions must be enabled)
`_ext4mptr`	-
`far, _far`	`__far`
`inline, _inline`	`#pragma inline[=forced|=never]`

near, _near	__near
restrict	-

Pragma directives	
`#pragma ADDRESS`	`#pragma location`
`#pragma ASM`	`asm, __asm`
`#pragma _ASMMACRO`	`asm, __asm`
`#pragma BIT`	-
`#pragma BITADDRESS`	-
`#pragma CREG` (V.6.00 only)	(M16C-specific)
`#pragma ENDASM`	-
`#pragma entry` (V.6.00 only)	`__noreturn` `__no_save`
`#pragma EXT4MPTR`	-
`#pragma INTCALL`	-
`#pragma INTERRUPT`	`#pragma vector = 0x17` `__interrupt void foo(void)`
`#pragma interrupt/V` (V.6.00 only)	`#pragma vector = 0x17` `__interrupt void foo(void)`
`#pragma ISTACKSIZE` (V.6.00 only)	(done via linker configuration file)
`#pragma JSRA` (V.5.45 only)	-
`#pragma JSRW` (V.5.45 only)	-
`#pragma PAGE`	-
`#pragma PARAMETER`	- (you must conform to the calling convention)
`#pragma ROM` (V.5.45 only)	`#pragma constseg`
`#pragma SBDATA`	-
`#pragma sectaddress` (V.6.00 only)	`#pragma segment|section`
`#pragma SECTION`	`#pragma location`
`#pragma SPECIAL`	-
`#pragma STACKSIZE` (V.6.00 only)	(done via linker configuration file)
`#pragma STRUCT`	`#pragma pack`
Intrinsic functions	
`abs_b`, `abs_w`	-
`dadc_b`, `dadc_w`	-
`dadd_b`, `dadd_w`	-
`div_b`, `div_w`	-
`divu_b`, `divu_w`	-
`divx_b`, `divx_w`	-
`dsbb_b`, `dsbb_w`	-
`dsub_b`, `dsub_w`	-
`mod_b`, `mod_w`, `modu_b`, `modu_w`	-
`movll`, `movlh`, `movhl`, `movhh`	-
`neg_b`, `neg_w`	-
`not_b`, `not_w`	-
`rmpa_b`, `rmpa_w`	-
`rolc_b`, `rolc_w`	-
`rorc_b`, `rorc_w`	-
`rot_b`, `rot_w`	-
`sha_b`, `sha_w`	-
`shl_b`, `shl_w`	-
`smovb_b`, `smovb_w`	-
`smovf_b`, `smovf_w`	-
`sstr_b`, `sstr_w`	-
Preprocessor symbols	
`__cplusplus` (V.6.00 only)	`__embedded_cplusplus`
`__DATE__`	`__DATE__`
`__FILE__`	`__FILE__`

`__LINE__`	`__LINE__`
`M16C`	`__ICCRL78__`
`NC30`	`__IAR_SYSTEMS_ICC__`
`__R8C__`	`__CORE__`
`__RENESAS__` (V.6.00 only)	`__IAR_SYSTEMS_ICC__`
`__RENESAS_VERSION__` (V.6.00 only)	`__VER__`
`__STDC__` (V.6.00 only)	`__STDC__` `__STDC_VERSION__`
`__TIME__`	`__TIME__`
Compiler options	
`-as30 <Option>`	-
`-c`	-
`-Didentifier`	`-D symbol[=value]`
`-dirdir_name`	`--output {filename|directory}` `-o {filename|directory}`
`-dsource` (`-dS`)	`-l[a|A|b|B|c|C|D][N][H] {filename|directory}`
`-dsource_in_list` (`-dSL`)	`-l[a|A|b|B|c|C|D][N][H] {filename|directory}`
`-E`	`--preprocess[=[c][n][l]] {filename|directory}`
`-exception` (V.6.00 only)	-
`-fansi`	`--strict`
`-fauto_128` (`-fA1`)	-
`-fauto_over_255` (`-fAO2`)	-
`-fbit` (`-fB`)	-
`-fchange_bank_always` (`-fCBA`)	-
`-fchar_enumerator` (`-fCE`)	`-e` (any integer type can be used)
`-fconst_not_ROM` (`-fCNR`)	-
`-fdouble_32` (`-fD32`)	- (default)
`-fenable_register` (`-fER`)	-
`-fextend_to_int` (`-fETI`)	-
`-ffar_pointer` (`-fFP`)	`__far` (keyword)
`-ffar_RAM` (`-fFRAM`)	-
`-finfo`	-
`-fjsrw` (V.5.45 only)	-
`-fnear_ROM` (`-fNR`)	`--data_model`
`-fno_align` (`-fNA`)	-
`-fno_carry`	-
`-fno_even` (`-fNE`)	-
`-fno_lib` (V.6.00 only)	-
`-fno_switch_table` (-fNST)	-
`-fnot_address_volatile` (`-fNAV`)	-
`-fnot_reserve_asm` (`-fNRA`)	-
`-fnot_reserve_far_and_near` (`-fNRFAN`)	-
`-fnot_reserve_inline` (`-fNRI`)	-
`-fptrdifft_16` (`-fP16`)	-
`-fSB_auto` (`-fSBA`)	-
`-fsizet_16` (`-fS16`)	-
`-fsmall_array` (`-fSA`)	-
`-fswitch_other_section` (`-fSOS`)	-
`-fuse_DIV` (`-fUD`)	`--disable_div_mod_instructions`
`-fuse_MUL` (`-fUM`)	-
`-g`	`--debug`
`-genter`	-
`-gbool_to_char` (V.5.45 only)	-
`-gno_reg`	-

`-gold` (V.5.45 only)	-
`-goptimize` (V.6.00 only)	-
`-I`*`directory`*	`-I` *`path`*
`-l`*`libraryfilename`*	(No option needed. Just add library name as input to the linker

`-lang={c|cpp|ecpp}` (V.6.00 only)	`-e`
	`--ec++`
	`--eec++`
	`--c89` (c99 is default)

`-ln30 <`*`Option`*`>` (V.5.45 only)	-
`-lnkcmd=<`*`filename`*`>` (V.6.00 only)	(available as a linker option, `-f`)
`-noexception` (V.6.00 only)	-
`-o`*`file_name`*	`--output {`*`filename|directory`*`}` `-o {`*`filename|directory`*`}`
`-O[1-5]`	`-Oh`
`-O5OA`	-
`-Ocompare_byte_to_word` (`-OCBTW`)	-
`-Oconst` (`-OC`)	-
`-Oforward_function_to_inline` (`-OFFTI`)	-
`-Oglb_jmp` (`-OGJ`) (V.5.45 only)	-
`-Oloop_unroll[=`*`unroll_count`*`]` (`-OLU`)	`#pragma unroll=`*`n`*
`-Ono_asmopt` (`-ONA`)	-
`-Ono_bit` (`-ONB`)	-
`-Ono_break_source_debug` (`-ONBSD`)	-
`-Ono_float_const_fold` (`-ONFCF`)	-
`-Ono_logical_or_combine` (`-ONLOC`)	-
`-Ono_stdlib` (`-ONS`)	-
`-OR`	`-Ohz`
`-OR_MAX` (`-ORM`)	`-Ohz`
`-OS`	`-Ohs`
`-OS_MAX` (`-OSM`)	`-Ohs`
`-Osp_adjust` (`-OSA`)	-
`-Ostack_frame_align` (`-OSFA`)	-
`-Ostatic_to_inline` (`-OSTI`)	-
`-P`	`--preprocess[=[c][n][l]] {`*`filename|directory`*`}`
`-preinclude=`*`filename`*`[…]` (V.6.00 only)	`--preinclude` *`includefile`*
`-R8C`, `-R8CE`	`--core={s1|s2|s3}`
`-rtti=off` (V.6.00 only)	(rtti is not supported)
`-rtti=on` (V.6.00 only)	(rtti is not supported)
`-S`	`-lb {`*`filename|directory`*`}`
`-silent`	`--silent`
`-U`*`predefined_macro`*	-
`-v`	(default behavior)
`-V`	Run the executable without options. Do not use `--silent`.
`-Wall`	`--no_warnings`
`-Wccom_max_warnings=`*`WarningCount`* (`-WCMW`)	`--error_limit=`*`n`*
`-Werror_file<`*`filename`*`>` (`-WEF`) (V.5.45 only)	`--diagnostics_tables {`*`filename|directory`*`}`
`-Wlarge_to_small` (`-WLTS`)	-
`-Wmake_tagfile` (`-WMT`) (V.5.45 only)	-
`-Wnesting_comment` (`-WNC`)	-
`-Wno_stop` (`-WNS`)	-
`-Wno_used_argument` (`-WNUA`)	-
`-Wno_used_function` (`-WNUF`)	-
`-Wno_used_static_function` (`-WNUSF`)	-
`-Wno_warning_stdlib` (`-WNWS`)	-

`-Wnon_prototype` (`-WNP`)	-
`-Wstdout` (V.5.45 only)	-
`-Wstop_at_link` (`-WSAL`)	ILINK option `--force_output`
`-Wstop_at_warning` (`-WSAW`)	-
`-Wundefined_macro` (`-WUM`)	-
`-Wuninitialize_variable` (`-WUV`)	-
`-Wunknown_pragma` (`-WUP`)	-

Assembler-specific details

Renesas	IAR Systems
Limitations in source code structure	
The assembler supports: * several modules per source file * relocatable placement	Same

Interrupt functions in assembler	

All interrupt vectors for peripheral interrupts should be placed in section `vector`, fixed interrupt vectors should be placed in section `fvector`.	Interrupt functions should be declared as `ROOT` so that they cannot be discarded by the linker even if no symbols in the segment are referred to. To insert an entry in the interrupt vector table, define the destination with the `DW` directive, for example like this:
	`COMMON INTVEC:CODE:ROOT(1)`
	`ORG 0x08 ;INTP0`
	`branchToInter0:`
	`DW inter0`

Sections are only defined using the assembler directive `.SECTION`. There is no distinction between absolute, relocatable, … sections. Section types can be `CODE`, `ROMDATA` or `DATA`.	Use the section control directive SECTION alias RSEG to place your code and data in sections. A section is *relocatable*.
`.SECTION` *name*`, CODE, [ALIGN]`	`RSEG` *section* `[:`*type*`] [:`*flag*`] [(`*align*`)]`
	Bit segments cannot be defined explicitly, but can easily be defined using bit operators in code or data segments. As a byte is the smallest allocatable memory segment, no memory is lost or gained using either tool.
Binary representation	
`00001111b` `00001111B`	Not supported, should be replaced by `0x0f`.

Renesas		IAR Systems
Integer constants		
`00001111B` `00001111b`	Binary	`1010b, b'1010`
`60702O` `60702o`	Octal	`1234q, q'1234, 01234`
`9423`	Decimal	`1234, -1, d'1234, 1234d`
`0A5FH` `0A5Fh`	Hexadecimal	`0FFFFh, 0xFFFF, h'FFFF`

Operand modifiers in assembler	
`:G, :Q, :S, :Z`	-
Assembler directives	
`?`	-
`@`	-
`.ADDR`	`DS24`
`.ALIGN`	`ALIGN`
`.ASSERT`	-
`.BLKA`	`DS24`
`.BLKB`	`DS, DS8`
`.BLKD`	`DS64`
`.BLKF`	`DS32`
`.BLKL`	`DS32`
`.BLKW`	`DS16`
`.BTEQU`	-
`.BTGLB`	-
`.BYTE`	`DS, DS8`
`.CALL`	-
`.DEFINE`	`#define, DEFINE`
`.DOUBLE`	`DS64`
`.EINSF`	`END`
`.ELIF`	`#elif`
`.ELSE`	`#else`
`.END`	`END`
`.ENDIF`	`#endif`
`.ENDM`	`ENDM`

.ENDR	ENDR
.EQU	EQU
.EXITM	EXITM
.FB	-
.FBSYM	-
..FILE	-
.FLOAT	DS32
.FORM, .LIST, .PAGE	LSTCND, LSTCOD, LSTEXP, LSTMAC, LSTOUT, LSTREP, LSTXRF
.GLB	PUBLIC, PUBWEAK
.ID	-
.IF	#if
.INCLUDE	#include
.INITSCT	RSEG
.INSF	NAME, PROGRAM
.INSTR	-
.LEN	-
.LOCAL	LOCAL
.LWORD	DS32
..MACPARA	_args
..MACREP	REPTC, REPTI
.MACRO	MACRO
.MREPEAT	-
.OFSREG	-
.OPTJ	-
.ORG	-
.PROTECT	-
.RESERVED_AREA (V.6.00 only)	DB, DC, DS
.RVECTOR	DC16 (see example in asm guide)
.SB	-
.SB_AUTO(_xxx)	-
.SBBIT	-
.SBSYM	-
.SECTION	RSEG
.STK	-
.SUBSTR	-
.SVECTOR	-
.VER	-
.WORD	DS16
Assembler options	
-.	--silent
-A	-
-C	-
-D	-D
-F	-
-finfo	-
-goptimize (V.6.00 only)	-
-H	-
-I	-I
-JOPT (V.5.45 only)	-
-L	-l
-M	-
-M60 (V.5.45 only)	--core
-M61 (V.5.45 only)	--core
-N	-le
-O	-o
-P	-

-PATCH(6N)_TA -PATCH(6N)_Tan	-
-R8C	--core
-R8CE	--core
-R8Cxx	--core
-S (V.5.45 only)	-l, --preprocess
-S[M] (V.6.00 only)	-l, --preprocess
-subcommand=<*filename*> (V.6.00 only)	-f
-T	--diagnostics_tables
-V	Run the executable without options. Do not use --silent.
-X	-

Linker and library details

Renesas	IAR Systems
Device-specific header files	
All SFR are defined in sfrxx.h and sfrxx.inc header files.	All SFRs are defined in ioxxx.h files.

Renesas	IAR Systems
Linker options	
-. (V.5.45 only)	--silent
@ (V.5.45 only)	--config <file>
Absolute_forbid (V.6.00 only)	-
Binary (V.6.00 only)	--image_input
BYte_count (V.6.00 only)	-
CAchesize (V.6.00 only)	-
CHange_message (V.6.00 only)	--diag_error, --diag_remark --diag_suppress, --diag_warning --diagnostics_tables --error_list, --no_warnings --remarks --warnings_are_errors --warnings_affect_exit_code
Compress, NOCOmpress (V.6.00 only)	-
CONTIGUOUS_SECTION (V.6.00 only)	Can only be specified in the linker configuration file.
CPu (V.6.00 only)	-
CRc (V.6.00 only)	(IAR ELF TOOL) --checksum {symbol[+offset]|address}:size,algorithm[:[1|2][m][L|W][r][i|p]][,start];range[;range...] Can be reserved with --place_holder *symbol*[,*size*[,*section*[,*alignment*]]]
DAta_stuff (V.6.00 only)	Can only be specified in the linker configuration file.
DEBug, Sdebug, NODEBug (V.6.00 only)	Debug information is included by default and removed by --strip. Debug information with terminal: --debug_lib
DEFine (V.6.00 only)	--define_symbol=*value*
DELete (V.6.00 only)	(IARCHIVE TOOL) --delete/-d libraryfile objectfile1 ... objectfileN
-E (V.5.45 only)	-
END (V.6.00 only)	-
ENTry (V.6.00 only)	--entry <symbol>
EXIT (V.6.00 only)	-
EXTract (V.6.00 only)	(IARCHIVE TOOL) --extract libraryfile [objectfile1 ... objectfileN] -x *l*ibraryfile [objectfile1 ... objectfileN]

`Form` (V.6.00 only)	-
`FSymbol` (V.6.00 only)	-
`FUnction_forbid` (V.6.00 only)	-
`-G` (V.5.45 only)	Automatic, if information is present in the input files.
`Hide` (V.6.00 only)	`--no_locals`
`-JOPT` (V.5.45 only)	-
`JUMP_ENTRIES_FOR_PIC` (V.6.00 only)	`Can only be specified in the linker configuration file.`
`-L` (V.5.45 only)	No separate option. Enter library names separated with blanks.
`-LD` (V.5.45 only)	`--search directory`
`LIBrary` (V.6.00 only)	No separate option. Enter library names separated with blanks.
`List` (V.6.00 only)	`--map {filename|directory}`
`-LOC` (V.5.45 only)	`Can only be specified in the linker configuration file.`
`LOgo`, `NOLOgo` (V.6.00 only)	Always output except in silent (`--silent`) mode
`-M` (V.5.45 only)	`--map {filename|directory}`
`-M60` (V.5.45 only)	(M16C-specific)
`-M61` (V.5.45 only)	(M16C-specific)
`MAp` (V.6.00 only)	`--map {filename|directory}`
`MEMory` (V.6.00 only)	-
`Message`, `NOMessage` (V.6.00 only)	-
`-MS` (V.5.45 only), `-MSL` (V.5.45 only)	`--map {filename|directory}`
`MSg_unused` (V.6.00 only)	-
`NOPRElink` (V.6.00 only)	-
`-NOSTOP` (V.5.45 only)	`--force_output`
`-O` (V.5.45 only)	`--output/-o {filename/directory}`

`Optimize` (V.6.00 only)	`--vfe=[forced]`
	`--merge_duplicate_sections`
	`--inline`

`-ORDER` (V.5.45 only)	`Can only be specified in the linker configuration file.`

`Output` (V.6.00 only)	`Only one output file. Extra formats can be generated via IAR ELF TOOL using the default output file as source.`
	(IAR ELF TOOL)
	`--bin`
	`--ihex`
	`--srec`
	`--simple`

`PADDING` (V.6.00 only)	-
`PROfile` (V.6.00 only)	-
`PS_check` (V.6.00 only)	Related to `-diag_supress`
`-R8C` (V.5.45 only)	-
`-R8CE` (V.5.45 only)	-
`REcord` (V.6.00 only)	-
`REName` (V.6.00 only)	`--redirect <from_sybol>=<to_symbol>`
`REPlace` (V.6.00 only)	(IARCHIVE TOOL) `--replace/-r`
`ROm` (V.6.00 only)	`Can only be specified in the linker configuration file.`
`RTs_file` (V.6.00 only)	-
`S9` (V.6.00 only)	-
`SAMECode_forbid` (V.6.00 only)	-
`SAMESize` (V.6.00 only)	-
`SBr` (V.6.00 only) (V.6.00 only)	-
`SEction_forbid` (V.6.00 only)	-

SHow (V.6.00 only)	--map {filename|directory}
SPace (V.6.00 only)	(IAR ELF TOOL) --fill [v;]pattern;range[;range...] (IAR ELF TOOL)
STACk (V.6.00 only)	-
STARt (V.6.00 only)	--entry <symbol>
STRip (V.6.00 only)	--strip
Subcommand (V.6.00 only)	--config <filename>
SYmbol_forbid (V.6.00 only)	-
-T (V.5.45 only)	-
Total_size (V.6.00 only)	Always except in -silent mode.
-U (V.5.45 only)	-
UTL (V.6.00 only)	-
-V (V.5.45 only)	Run the executable without options. Do not use --silent.
Variable_forbid (V.6.00 only)	-
-VECT (V.5.45 only)	By default, the vector table is populated with a *default interrupt handler* which calls the abort function. For each interrupt source that has no explicit interrupt service routine, the default interrupt handler will be called. If you write your own service routine for a specific vector, that routine will override the default interrupt handler.
VECT (V.6.00 only)	See above.
-VECTN (V.5.45 only)	See above.
VECTN (V.6.00 only)	See above.
-W (V.5.45 only)	--diag_error, --diag_remark --diag_suppress, --diag_warning --diagnostics_tables --error_list, --no_warnings --remarks --warnings_are_errors --warnings_affect_exit_code
Segments/Sections	
bss	.bss.noinit .bssf.noinit
C$INIT (V.6.00 only)	The section name is not relevant. ILINK looks at the segment type to recognize these.
C$VTBL (V.6.00 only)	The section name is not relevant. ILINK looks at the segment type to recognize these.
data	.data .dataf .hdata .sdata
fvector	.intvec
heap (V.5.45 only)	NEAR_HEAP FAR_HEAP HUGE_HEAP
heap_NE (V.6.00 only)	NEAR_HEAP
interrupt	-
istack (V.6.00 only)	CSTACK
program	.text .textf
program_S	Not fixed. Can be forced to a user defined segment with #pragma constseg or #pragma dataseg
rom	.const .constf

	`.consth`
`stack`	`CSTACK`
`switch_table`	`.switch` `.switchf`
`vector`	`.intvec`

Runtime environment

Renesas	IAR Systems
Calling convention	
Parameters passed on the stack	
all except char, short, near pointer	
Parameters passed in registers	
8-bit values in: `R1L`	8-bit values in: `A`, `B`, `C`, `X`, `D`, `E`
16-bit values in: `R1/R2`	16-bit values in: `AX`, `BC`, `DE`
24-bit values in:	24-bit values in: Stack
32-bit values in: stack	32-bit values in: `BC:AX`
Floating-point values in: stack	Floating-point values in: `BC:AX`
Return values	
8-bit values in: `R0L`	8-bit values in: `A`
16-bit values in: `R0`	16-bit values in: `AX`
24-bit values in:	24-bit values in: `A:HL`
32-bit values in: `R2R0`	32-bit values in: `BC:AX`
64-bit values in: `R3R1 R2R0`	
Floating-point values in: `R2R0`	Floating-point values in: `BC:AX`
Preserved registers	
None	`BC` and `DE`
Scratch registers	
-	-The registers `AX`, `HL`, `CS` and `ES`. -Registers that are used as register parameters and for returning values by a function.
System startup and exit code	
The system startup code can either be delivered in assembler files (`ncrt0.a30` + `sect30.inc`) or in C source files.	The system startup code is located in the ready-made `cstartup.s` file. In addition, you specify additional settings, for example for the stack and heap size. It is likely that you need to customize the code for system initialization. For example, your application need to initialize memory-mapped special function registers, or omit the default initialization of data segments performed by cstartup. You can do this by providing a customized version of the routine `__low_level_init`, which is called from cstartup before the data segments are initialized. Modifying cstartup directly should be avoided.
Global variable initialization	
Static and global variables are initialized: not-initialized variables are cleared and the values of other initialized variables are copied from ROM to RAM memory.	Static and global variables are initialized: zero-initialized variables are cleared and the values of other initialized variables are copied from ROM to RAM memory. This initialization can be overrided by returning 0 from the `__low_level_init` function. Variables declared `__no_init` which are not initialized at all: `__no_init int i;`
Reentrancy and recursive functions	
Not all library functions are reentrant. Refer to Appendix E of NC30 User's Manual for details.	The compiler is always reentrant when using the DLIB library.