ARM SDT 2.50 User and Reference Guides **Errata 01**

This errata document gives details of documentation errors in the ARM SDT 2.50 *User Guide* and *Reference Guide*. It does not list the bug fixes made for the SDT 2.51 release. Refer to the SDT 2.51 readme document for more information.

Note

SDT 2.51 is a maintenance release of the SDT 2.50 toolkit. SDT 2.51 does not introduce any additional functionality. The documentation for SDT 2.50 remains current for the SDT 2.51 release.

1 ARM SDT 2.50 User Guide - ARM DUI 0041D

This section gives details of documentation errors in the SDT 2.50 *User Guide*. It also provides additional information on using the Gateway DLL with ADW. See *Additional information for the gateway DLL* on page 4 for more information.

1.1 Text corrections

Table 1 User Guide text corrections

Page	Para	Current text /Problem	Replacement text/Correction
ix	4th	the version string of the tool, including the version number and date.	the version number of the tool, including the version number and build number.
6-6	3rd	The example code in example 6-1 is incorrect.	Ignore the example. The example is redundant, except as an illustration of the APCS, because SDT 2.50 and above support **`long long`**.

9-18	1st	In example 9-8, r12 is an APCS callee-corruptible register, so typically you must store it on entry to an exception handler. Lines 1 and 10 of the code example are incorrect.
		`STMFD sp!, {r0-r3,lr}`
		`...`
		`LDMFD sp!, {r0-r3,pc}^`
			The corrected lines are:
			`STMFD sp!, {r0-r3,r12,lr}`
			`...`
			`LDMFD sp!, {r0-r3,r12,pc}^`

9-33	1st	The context switch code in example 9-17 is incorrect.	The correct text is:
			`STMIA    r13, {r0-r14}^`
			`MRS      r0, SPSR`
			`STMDB    r13, {r0,r14}`
			`LDR      r13, [r12], #4`
			`CMP      r13, #0`
			`LDMNEDB  r13, {r0,r14}`
			`MSRNE    spsr_cxsf, r0`
			`LDMNEIA  r13, {r0-r14}^`
			`NOP`
			`SUBNES   pc, lr, #4`

Table 1 User Guide text corrections

Page	Para	Current text /Problem	Replacement text/Correction

10-19	3rd	In section 10.5.1 step 3, the second command line is incorrect:
		`armlink -o tram0.axf`
		`-ro-base 0xf0000000`
		`-ro-base 0x10000000`
		`-First init.o(Init)`
		`-map -info Sizes init.o ex.o`
			The correct command line is:
			`armlink -o tram0.axf`
			`-ro-base 0xf0000000`
			`-rw-base 0x10000000`
			`-First init.o(Init)`
			`-map -info Sizes init.o ex.o`

10-27 and 10-30	1st and 4th	The example makefiles refer to: `C:\ARM250\lib\embedded\armlib_cn.321`	The correct library file name is: `armlib_cn.321`
12-33	last	The version of Microsoft Visual C++ Developer Studio required to rebuild the ARMulator is incorrect.	To rebuild the ARMulator, load `armulate.mak` into Microsoft Visual C++ Developer Studio (version 5.0 or greater).
13-84	4th	The return value for an interactive device is incorrect.	On exit, r0 contains: • 1, if the handle identifies an interactive device • 0, if the handle identifies a file • a value other than 1 or 0 if an error occurs.
13-90	2nd	In this case, the values of the following symbols are used:	In this case, the following symbols point to words of data containing the locations:

13-90	4th	On entry, r1 points to a single word data block	On entry, r1 must contain the address of a pointer to a four word data block.
			Word 1 does not need to contain a value.
			Example:
			`struct block2 {`
			`             int heap_base;`
			`             int heap_limit;`
			`             int stack_base;`
			`             int stack_limit;`
			`                       };`
			`struct block2 *mem_block, info;`
			`mem_block = &info;`
			`SemiSWI(SYS_HEAPINFO,`
			`        (unsigned)&mem_block);`

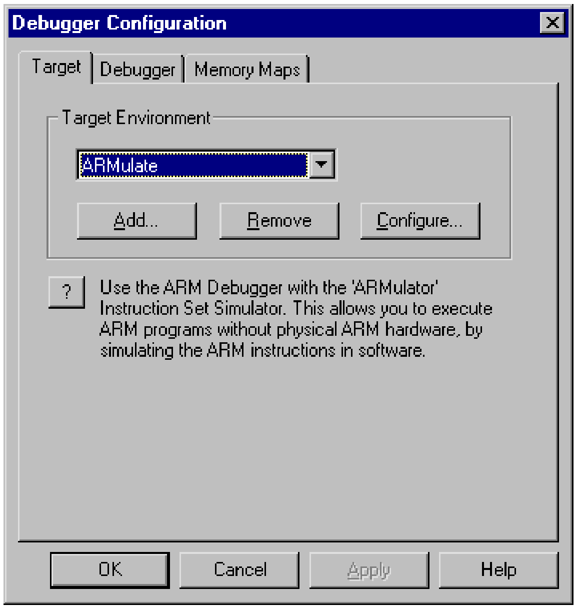
1.2 Additional information for the gateway DLL

SDT 2.51 includes an upgraded version of the gateway.dll that provides:

- improved download speed
- support for the Debug Communications Channel (DCC)
- support for the latest ARM7-series cores (ARM710T, ARM720T, ARM740T) in addition to ARM7TDMI and ARM7DI.
- gateway configuration files for automatic input of coprocessor register descriptions

To install the Gateway DLL:

1. Start ADW.
2. Select **Configure Debugger…** from the **Options** menu. The Debugger Configuration panel is displayed (Figure 1).



Target Environment: ARMulate

Use the ARM Debugger with the 'ARMulator' Instruction Set Simulator. This allows you to execute ARM programs without physical ARM hardware, by simulating the ARM instructions in software.

Figure 1 Debugger Configuration panel

3. Click **Add…**. A standard file dialog is displayed (Figure 2)

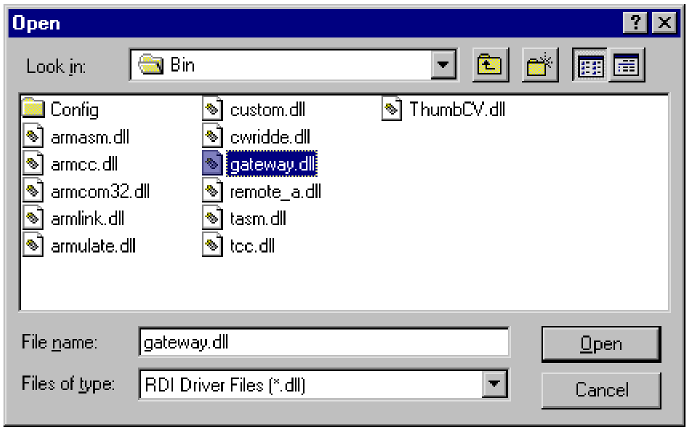


Figure 2 Adding the gateway dll

4. Select `gateway.dll` and click **Open**.
5. Click **Configure…** in the Debugger Configuration screen. The Gateway Remote Configuration panel is displayed (Figure 3).

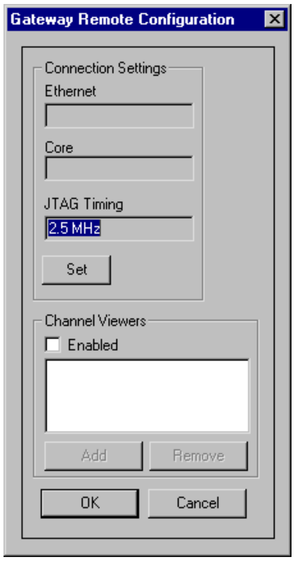


Figure 3 Gateway Remote Configuration panel

6. Click **Set** to set connection details for your probe. A Set Connection Details panel is displayed (Figure 4).

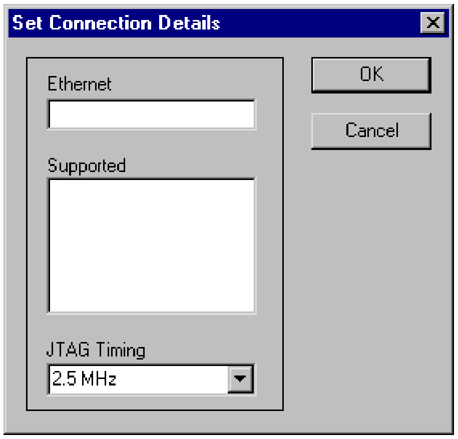


Figure 4 Set Connection Details panel

7. Enter the IP address for your probe in the Ethernet field. See your HP documentation for more information on setting the IP address.

8. Press the Tab key, or click in the Supported field. The Debugger establishes a connection with probe and displays a list of supported targets in the Supported field.

9. Select the target type you want in the Supported field.

10. Select the JTAG base clock speed you require from the JTAG Timing drop-down menu. This sets the frequency at which the probe clocks data across the target JTAG port. Higher frequencies give improved performance, especially for JTAG-intensive operations such as downloading.

    You are recommended to select the highest frequency supported by your target hardware. Hardware constraints such as stacking several devices together, or using long cables between the probe and the target, might require you to use lower JTAG frequencies.

11. Click **OK** to confirm your settings and close the Set Connection Details panel.

12. Click **OK** in the Gateway Remote Configuration panel.

13. Click **OK** in the Debugger Configuration screen to restart the debugger with a connection to the probe.

2 ARM SDT 2.50 Reference Guide - ARM DUI 0040C

This section gives details of documentation errors in the SDT 2.50 *Reference Guide*.

2.1 Text corrections

Table 2 Reference Guide text corrections

Page	Para	Current text /Problem	Replacement text/Correction
ix	4th	the version string of the tool, including the version number and date.	the version number of the tool, including the version number and build number.
2-21	5th	The following `-arch` options are documented but are not supported by the compilers: • `-arch 4M` • `-arch 4xM`	Ignore the options.

2-30	5th	The code example for the description of -Wg is incorrect:
		`#ifdef foo_h`
		`#define foo_h`
			`#ifndef foo_h`
			`#define foo_h`

2-32	3rd	For C code, suppresses warnings about future compatibility with C++ for both armcpp and tcpp. This option is off by default. It can be enabled with `-W+u`.	For C code, `-Wu` suppresses warnings about future compatibility with C++. Warnings are suppressed by default. They can be enabled with `-W+u`.
3-50	1st	Table 3-16 Minor language feature support, is incomplete.	Insert the following table entry after the entry for **`wchar_t`**: **Minor Language Feature** **Supported** **`mutable`** keyword No

Table 2 Reference Guide text corrections

Page	Para	Current text /Problem	Replacement text/Correction

4-10	3rd	The C library initialization code in example 4-1 on page 4-10 is incorrect.	The correct text and example code is:
			To initialize the libraries you must call `__rt_lib_init` with four parameters. The first three are currently unused, and so can be zero. The fourth should point to a C++ init block, containing the addresses of `__cpp_initialise` and `__cpp_finalise`.
			`__cpp_initialise` and `__cpp_finalise` only exist when linking with C++ code, therefore they are weak references. When linking with C code, they take the value zero.
			
			`AREA LibInit, CODE, READONLY`
			
			`   IMPORT __rt_lib_init`
			`   IMPORT __cpp_initialise, WEAK`
			`   IMPORT __cpp_finalise, WEAK`
			
			`   EXPORT init_library`
			
			`init_library`
			`    MOV      r0,#0`
			`    MOV      r1,#0`
			`    MOV      r2,#0`
			`    ADR      r3, CppInitBlock`
			
			`    B        __rt_lib_init`
			
			`; Return to the caller...`
			
			`CppInitBlock`
			`    DCD      __cpp_initialise`
			`    DCD      __cpp_finalise`
			
			`    END`

5-9	2nd	The list of predeclared register names is incomplete.	The following register names are reserved for future use: s0-s31 S0-S31

Table 2 Reference Guide text corrections

Page	Para	Current text /Problem	Replacement text/Correction
5-14 and 5-15	6th and 6th	The offset from the pc to the constant must be less than 4KB.	The offset from the pc to the constant must be less than 1KB.
6-8	5th	The description of the `-noremove` option is does not state that `-noremove` is the default.	`-noremove` is the default.
6-9	1st, 2nd and 3rd	The descriptions for the `-entry`, `-first`, and `-last` options are incomplete.	Add the following note to each description: **Note** On UNIX systems you might need to escape the parentheses characters with a backslash (`\`) character.
7-36	4th	If no `/format` string is entered, integer values default to the format described by the variable `$format`	If no `/format` string is entered, integer values default to the format described by the variable `$int_format`.
7-56	last	If no format string is entered, integer values default to the format described by the variable `$format`.	If no format string is entered, integer values default to the format described by the variable `$int_format`.
8-4	last	The description of the fromELF output file format is incomplete.	ELF images will contain multiple load regions if, for example, they are built with a scatter load description file that defines more than one load region. If you use fromELF to convert an ELF image containing multiple load regions to a binary format using any of the `-bin`, `-ihf` `-m32`, or `-i32` options, fromELF creates an output directory named *`output_file`* and generates one binary output file for each load region in the input image. fromELF places the output files in the *`output_file`* directory.
8-10	3rd	The `decaof` command options are incomplete.	The following additional options are supported: `-m` display mangled names. `-only` *`symbol_name`* process only the area named, or containing the symbol *`symbol_name`*.

Table 2 Reference Guide text corrections

Page	Para	Current text /Problem	Replacement text/Correction
11-2	6th	The reference to the ARM FPA 10 datasheet is incorrect.	The ARM FPA 10 datasheet is obsolete, and is no longer available.
11-13	1st	For information on how to configure the FPASC for a new environment, see Application Note 10: *Configuring the FPA Support Code/FPE* (ARM DAI 0040)	For information on how to configure the FPASC for a new environment, see Application Note 40: *Configuring the FPA Support Code/FPE* (ARM DAI 0040)
15-15	1st	refer to 3.6, Handling Relocation Directives on page 3-16.	refer to 6.13 *Handling Relocation Directives* on page 6-41.

2.2 Tables

The following corrected tables replace Table 3-2 on page 3-15, 4-4 on page 4-21, and Table 7-4 on page 7-54. Changed lines are identified by change bars.

Table 3-2 Escape codes

Escape Sequence	Char value	Description
\a	7	Attention (bell)
\b	8	Backspace
\t	9	Horizontal tab
\n	10	New line (linefeed)
\v	11	Vertical tab
\f	12	Form feed
\r	13	Carriage return
\xnn	0xnn	ASCII code in hexadecimal
\nnn	0nnn	ASCII code in octal

Table 4-4 Supported C library functions

File	Functions				
math.h	acos	asin	atan	atan2	ceil
	cos	cosh	exp	fabs	floor
	fmod	frexp	ldexp	log	log10
	modf	pow	sin	sinh	sqrt
	tan	tanh			
stdlib.h	abs	atoi	atol	atof	bsearch
	calloc	div	free	labs	ldiv
	malloc	qsort	realloc	strtod	strtol
	strtoul				
ctype.h	isalnum	isalpha	iscntrl	isdigit	isgraph
	islower	isprint	ispunct	isspace	isupper
	isxdigit	tolower	toupper		
string.h	memchr	memcmp	memcpy	memmove	memset
	strncpy	strncmp	strcat	strcmp	strcpy
	strlen	strchr	strcspn	strncat	strrchr
	strspn	strstr	strxfrm	strpbrk	
stdio.h	sprintf	sscanf	vsprintf		
setjmp.h	setjmp	longjmp			

Table 7-4 armsd variables

Variable	Description
`$clock`	number of microseconds since simulation started (ARMulator only). This variable is read-only. This variable is read only, and is only available if a processor clock speed has been specified (See the ARM Software Development Toolkit User Guide for information on how to specify the emulated processor clock speed)
`$cmdline`	argument string for the debuggee.
`$echo`	non-zero if commands from obeyed files should be echoed (initially set to 01).
`$examine_lines`	default number of lines for the examine command (initially set to 8).
`$int_format`	default format for printing integer values (initially set to 0x%.8lx).

Table 7-4 armsd variables (continued)

Variable	Description
`$float_format`	default format for printing floating-point values (initially set to %g").
`$uint_format`	Default format for printing unsigned integer values (initially “0x%.8lx”).
`$sbyte_format`	Default format for printing signed byte values (initially “%c”).
`$ubyte_format`	Default format for printing unsigned byte values (initially “%c”).
`$string_format`	Default format for printing string values (initially “%s”).
`$complex_format`	Default format for printing complex values (initially “(%g,%g)”).
`$fpresult`	floating-point value returned by last called function (junk if none, or if a floating-point value was not returned). This variable is read-only. `$fpresult` returns a result only if the image has been built for hardware floating-point. If the image is built for software floating-point, it returns zero.
`$inputbase`	base for input of integer constants (initially set to 10).
`$list_lines`	default number of lines for `list` command (initially set to 16).
`$memory_statistics`	outputs any memory map statistics which the ARMulator has been keeping. This variable is read-only. See the ARM Software Development Toolkit User Guide for further details.
`$rdi_log`	rdi logging is enabled if non-zero, and serial line logging is enabled if bit 1 is set (initially set to 0).
`$arm_swi`	the ARM semihosting SWI number.
`$thumb_swi`	the Thumb semihosting SWI number.
`$pr_linelength`	sets the display width of armsd output.
`$result`	integer result returned by last called function (junk if none, or if an integer result was not returned). This variable is read-only.
`$sourcedir`	directory containing source code for the program being debugged (initially set to the current directory).

Table 7-4 armsd variables (continued)

Variable	Description
`$statistics`	outputs any statistics which the ARMulator has been keeping. This variable is read-only.
`$statistics_inc`	similar to `$statistics`, but outputs the difference between the current statistics and those when `$statistics` was last read. This variable is read-only.
`$top_of_memory`	This is used to enable Multi-ICE, EmbeddedICE, and Angel, and to return sensible values when a `HEAP_INFO SWI` call is made to determine where the heap and stack should be placed in memory. The default is 0x80000 (512KB). This should be modified before executing a program on the target if the memory size available differs from this.
`$type_lines`	default number of lines for the `type` command.
`$vector_catch`	indicates whether or not execution should be caught when various conditions arise. The default value is `%RUsPDAifE`. Capital letters indicate that the condition is to be intercepted: `R` reset `U` undefined instruction `S` SWI `P` prefetch abort `D` data abort `A` 26-bit address (Reserved, do not use) `I` IRQ `F` FIQ `E` error (Reserved, do not use).