



1 VersaKit-30xx contents:

The VersaKit-30xx development kit is a plug-and-play evaluation system for the VRS51L3xxx series of high performance, fully-integrated, FRAM-Enhanced[™] 8051 microcontrollers (MCUs). The VersaKit-30xx provides a complete and comprehensive programming and development platform, with ample prototyping space and easy access to chip peripherals and I/Os.

The VersaKit-30xx ships complete with:

- o Development board that supports the VRS51L3074 (or VRS51L2070)
- VJTAG-USB programming/debugging interface
- o USB cable
- 6V power supply (Euro or U.S. compatible)



Figure 1: Complete VersaKit-30xx kit

2 Software and Documentation

The VersaKit-30xx Getting Started Guide is a quick reference guide to setup the VersaKit-30xx.

Before you begin, visit the Ramtron web site at <u>http://www.ramtron.com/doc/Products/Microcontroller/Support_Tools.asp</u> to download the following:

- VersaKit-30xx User Guide
- Versa Ware JTAG software

The user guide provides a complete description of the VersaKit-30xx feature set, as well as the development board schematics. The Versa Ware JTAG Software provides an easy-to-use platform for device programming and in-circuit debugging of the code. Include files and demonstration programs for the VRS51L3074 can be downloaded from the abovementioned web page.

Download the VRS51L3074 datasheet at http://www.ramtron.com/doc/Products/Microcontroller/MCU_list.asp.

٠

1-800-545-FRAM, 1-719-481-7000

[♦] MCU customer service: 1-800-943-4625, 1-514-871-2447 x 208

3 Overview: VersaKit-30xx Development Board and the VJTAG Interface

The figure below presents an overview of the VersaKit-30xx development board and its principal features:



Figure 2: VersaKit-30xx development board

The VJTAG-USB board is a USB-based JTAG interface that enables in-circuit programming and debugging of the Ramtron VRS51L3074 and other JTAG-based MCUs. The VJTAG-USB connects directly to the VersaKit-30xx development board and can be used on prototype/production boards featuring any Ramtron JTAG-based MCU.





4 Basic Development Board and VJTAG Setup

The setup of the VersaKit-30xx is quick and easy. However, the order in which the operations are performed is important and should be followed.

First, insert the VJTAG-USB into Header H2 on the devboard so that the CN1, USB connector sits directly on top of the devboard's rightmost DB9 connector. If the VJTAG-USB is to provide power to the devboard, then a jumper should be inserted at position JP1. If the development board's is to have an external power supply, the jumper should be removed from JP1. Consult the VersaKit-30xx for more details on power supply configuration.



Figure 4: Installing the VJTAG-USB on the devboard

Second, connect the USB cable to the VJTAG-USB board.



Figure 5: Connecting the USB cable to the VJTAG-USB

Form this point, it is important not to remove the VJTAG-USB board from the development board without first disconnecting the USB cable. Doing so may cause Windows® OS instability, and require a computer reboot.

5 Overview: Versa Ware JTAG Software

Versa Ware JTAG is a Windows®-based software tool that provides a user-friendly development platform for all Ramtron microcontrollers featuring a JTAG interface (the VRS51L2xxx, VRS51L3xxx and future derivatives).

The Versa Ware JTAG Software is composed of two parts:

• Versa Ware JTAG Programmer

The Versa Ware JTAG Programmer is used to perform operations such as erase, program, read, etc., on the target device's Flash memory.

• Versa Ware JTAG Debugger

The Versa Ware JTAG Debugger is a user interface that links the in-circuit debugger and the source code. All Ramtron MCUs with a JTAG interface include an integrated debugger that enables in-application debugging of the device via its JTAG interface.

The Versa Ware JTAG Debugger is compatible with the SDCC, Keil and Ride compilers.

The Versa Ware JTAG Software has been developed under Windows XP, but should also run on Windows Vista and Windows 2000 operating systems, as well as Window 98 SE Operating Systems, as long as the PC has a UBS interface. Systems without a USB interface can use Ramtron's parallel port version of the JTAG interface. Please contact Ramtron for details.

The following window shows the in-circuit debugger in action when the source code XRAM, SFR page 0 and watch list windows are open and the program is halted at a breakpoint.

🛃 Versa Ware JT/	AG - YourCode_SDCC.	ihe .										
Action Edit View	Breakpoint Watch)	Eindow Help										
	P5 🔊 🔥 🔟 🖪	S 🕱 🗉 🗉 😾 😾 🖞	2 98 🐀 🖾 🖸 🖸) 🔊 🔁 🖼 🛙	🗄 📾 🔓	6 4	4, 5					
🔏 🔁 🔶 💓												
BPO: Program 0x005	59 BP1 not set	BP2 not	set BP3 (vot set	BP4 not	set		BPS not se	e.			
R YourCode SD	C.bx				RXRAM		_		_		- 10	
(55)	// Send 'EXT	DITS Received' on 10	8T0	A100,25		0	1	2	2		1	6 .
. (56)	char entr = 0x0	.	// Init outr to m	at to pessage	000	0.45	0.45	0.63	0.40	0.0E	0.00	
(57)	DUTINA = 0x00:		//Disable Interry	nt co neurope		UNVIE 0	0.40	0.61	OKIC	0.00	0.40	-
0050	BOY	INTEN1 . #0x00	//Parameter incenter		010	UKJC	040	UAL1	UM61	0409	0x19	0.
. (58)					020	0x8D	0x02	0x00	0x0C	0x21	0.42	0:
	4.1			_	030	0x86	0x40	0x79	0xDF	0xC3	0x8A	Q:
8- 0 [33]	aot	-3 40-00			040	0x9E	0.64	0x4B	0x45 (DACD .	0x48	Û:
- 00053	BOV	E2,#0800			050	OVEE	0+4E	0,94	0w4C	0-49	DIFC	0.
- 0055	001010:				050	0.67	0.19	0.00	0.09	0.45	0.21	-
8- 🄶 [ee]	B = negint0	[cptx++];			000	URE 7	0815	UNDO	0.03	0.00	0.40	-
- 0 0055	nov	ar3,r2			070	0x76	0.08	0x21	0M07	OMO7	0x10	0.
- 0057	inc	£2			- m	0.61	0-07	0.06	0-04	NCS I	nuen I	ΩŤ
- 0058	207	a,r3										1
- 🥝 0059	807	dptr,f_msgint0			🕂 SFR Pag						- 1	
005C	BOVC	a,8a+dptr			Nam	. 1	Address	Bin	Dec	Hex	ASCI	12
- 0 005D	BOV	B.a			P0		0,00	1111 1111	- 4	0.FF	U U	11
- · [61]	tamit@(B):				50	_	0,01	0000.0111		0,02	0.01	-11
005F	809	dpl, B			- or	_	0.01	000000111		0.07	DEL	-11
0062	nush	ar2			DPU	,	0x05	0000 0000	0	0800	NUL	-11
0064	Icall	typit0			DPH	0	0x03	0000 0000	0	0x00	NUL	
0.0062	non				DPL:	1	0x04	0000 0000	0	0x00	NUL	
	hubile (mori	ano			DPH	1	0.05	0000 0000	0	0,00	NUL	1
B- • [62]	yaarreinagr	scaleberli (s.):			DPS		0,96	0000 0000	0	0,00	NUL	1
0069	BOV	0,52 data 4 analas0			800		0.07	0110.0000		0.60	1404	-11
- 000A	BOV	dper,#_msgines		-1	1 100	•	UNDY	0110 0000	20	0400	1.0.0	-11
- 0 006D	BOAC	a,8a+dptr	_		INTER	n	0x88	0000 0000	0	0.00	NUL	-14
N				10	1 10110	n I	0,09	0000.0000	1 0	I n.n	I MIT	12
Name		Scope	Type	Memory		Size	1	Address	1	Vak		
🔔 cptr		INTOInterrupt81:L1	char	IRAM		1		0x02		1	1.	-
🔔 cptr		INT1InteruptB1:L1	char	IRAM		1		0x02		1	1'	
genvar		UARTOInterrupt B1	char	IRAM		- 1		0x03		1	a.	
cptr		main(U1)L1	PV/TE	IDAM .		2		0x02			1	
A disisten		delay 8111	LINT	1DAM		- 1		0.04		UM	1	
A varback		Gisbal 8010	char	VELAN		1		0-0010		5.1	7	-
- recreat		00000000	17.00					080010				_
realipoint 0 has been	n reached								0x0059 L	ine 60, Ad	dress 0x8	3059

Figure 6: Versa Ware JTAG Debugger window

The Versa Ware JTAG Debugger provides a comprehensive set of configuration options allowing the user to tailor the user interface, the watch variables and the breakpoints settings.

6 Installing and Using the Versa Ware JTAG Software

Installation of the Versa Ware JTAG Software is handled automatically by the set up program, which installs the:

- Versa Ware JTAG Software
- Prolific PL2303X USB driver

As is the case with most software/driver installations, we recommend creating a system restore point before running the setup. To proceed with Versa Ware JTAG installation, download from the Ramtron web site and run the following file: Versa_Ware_JTAG_3x_SETUP.exe.

Once the software is installed, it can be run directly from the setup program or by clicking on the Versa Ware JTAG shortcut created during the installation process.



Figure 7: Versa Ware JTAG icon

6.1 Versa Ware JTAG Programming Interface

Upon startup, the software will attempt to connect to the VJTAG-USB interface.

🔁 Yersa Ware JTAG - YourCode_SDCC.ihx		_ 🗆 ×
<u>File Flash</u> <u>D</u> evice <u>View</u> <u>W</u> indow <u>H</u> elp		
l 🚍 🚰 🦨 🖉 🐚 🕂 🖕 🛖 🞞 🕨	• 🛞 📊 🛈	
	- + +	
Action Toolba		
- File		
Name: YourCode_SDCC.ihx	File Information Window	
Size: 750 Bytes		
CRC: 2A27CF80	Info	
File passes integrity check.	Erase Prog	
Status Bar		
VR551L3074-40-Q waiting for instruction		AAX DBG VER //.

Figure 8: Versa Ware JTAG programming interface

Most of the functions provided by the Versa Ware JTAG Software are accessible through the action toolbar.

To download a HEX file into the VRS51L3074:

- 1. Make sure that the Versa-JTAG interface is properly connected to the H2 header of the devboard.
- 2. Click on the Synchronize button. The status bar should show: "VRS51L3074-40-Q waiting for instruction".
- 3. Click Open to select the HEX file to be programmed into the VRS51L3074.
- 4. Click on Erase then Program $\stackrel{\textcircled{2}}{\bullet}$ to erase and program the Flash. By default, after this process is complete, the program will start.

The Synchronize button can be used to halt execution of the VRS51L3074 program and put the device into program mode. The Run button restarts program execution.

The Options **1** button allows configuring the programming options, set the Flash security options, and activates the incircuit debugger.

Device Options	×
Boot-up Clock Divider: Off FRAM is read only	Verification C Verify entire flash C Verify programmed file only C Do not verify (not recommended)
Programming Enable debugging Automatic power control Run device after programming Close previous file when opening Display message when complete	Security Do not secure flash Secure flash this time only Always secure flash Warn when removing security Warn when secured device is found
MCU: VR551L3074-40-Q	OK Cancel

Figure 9: Versa Ware JTAG device options

Changes to any device options will become effective the next time an Erase then Program vore program vore to any device options will become effective the next time an Erase then Program vore the transformation of tra

6.2 Using the Versa Ware JTAG Debugger

Once the program is loaded into the VRS51L3074 Flash memory and the debugger is enabled, activate the debugger by clicking the Debugger button. Upon startup, the debugger will halt the processor at address 0x0000 and wait.

🛃 Yersa Ware JTAG - YourCode_SDCC.ihx									
Action Edit View Breakpoint Watch Window Help									
🔼 🔲 🗋 🚺 🖡	🖇 🔁 😓 🗖 🛐	🗙 II F 📆 🐯 💆	15 🐀 😰 🖸 🔜	N 🗧 🗄 🖽	🛗 🔓 🎑 🚜 🦧	5 E ,			
🧟 🐚 🛧 🔞		,,,	·						
BPO not set	BP1 not set	BP2 not se	t BP3 not	set	BP4 not set	BP5 not set			
	c 1]							
TourLode_SDL	Linx								
C: \Your_cod	e \YourCode_SDCC.C			-					
0000	interrupt_ve	ct:							
	ljmp	sdcc_gsinit_startup							
0003	ljmp _I	NTOInterrupt							
0006	.ds 5								
000B	ljmp _I	NTlInterrupt							
000E	.ds 5								
0013	reti								
0 0014	.ds 7								
001B	reti								
001C	.ds 7								
0023	reti								
0023	40.7								
0024	.us /								
UU2B	тјшр _о	ARIUINTErrupt							
[] [01] ///.	,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,								
[[02] // YourCode_SDCC.c //									
■ ■ [04] //									
- • [05] // S	This program initi	alise the UARTO at 1	15200 (with SOSC = 4	10MHz)					
[] • • [06] // :	It then transmit "	UARTO Echo: Waiting	for char on RXD0	.or INT0\0" (
• • [07] // S	The UART, INTO and	INT1 are configured	and the program end	ters in infi					
08] // j	As soon as a chara	cter is received it i	is transmitted back	on TXD0					
09] // :	If INTO or INT1 is	received the program	n transmit "EXT INT) received" (
[0 [10] // 0	on TXD0 depending	on which interreupt w	mas received.	-					
X Name		1.0	france fr	·	ci [A.44	Valuel II		
		INTOInterrupt: P1:11	l Type M	Iemory	5ize				
		INT1Interrupt:B1:L1	char If	RAM	1	0x02	'\3'		
A genvar		UART0Interrupt:B1:	char If	RAM	1	0x03	, '\0'		
🔺 optr		main:B1:L1	int If	RAM	2	0x02	3		
<u>_</u> <u>A</u> x		delay:B1:L1	BYTE	RAM	1	0x04	0x00		
📓 🔺 dlaisloop		delay:B1:L1	UINT IF	RAM	2	0x02	3		
A varcheck		Global:BU:LU	cnar ×	HAM	1	UXUUTU	Ë 🗾		
Processor halted						0×0000			

Figure 10: Versa Ware Debugger upon startup

Once the debugger has started, either run the program at full speed by clicking on the Run \square button or set a breakpoint anywhere in the code by either double clicking on a specific code line and then clicking the Run \square button or by setting the breakpoint manually. Alternatively, you can manually set breakpoints anywhere in the code by clicking the Breakpoint Setting button.

When a breakpoint is set in the code, the breakpoint reference number will be displayed in red next to the code line. The breakpoint toolbar will then display the active breakpoint. Double clicking again on a given code line where a breakpoint has been set will disable that breakpoint.

🛃 Versa Ware JTA(G - YourCode_SDCC.ihx		
<u>Action Edit View</u>	<u>B</u> reakpoint Watc <u>h W</u> indow (Help	
🗖 🖬 🗋 🚺 🦷	5 🥥 🧶 🎞 S 🗙 🖬	I 🗉 👿 🕱 💆 📽	1 🔁 🛛 🖸 🔁 🗄
la 🔁 🔁 🦓			
BP0: Program 0x016E	BP1 not set	BP2 not set	BP3 not set
R YourCode_SDC	Lihx		
⊡ □ [131] vo	id main (void){		
💷 😐 016E	_main:		
. [132]	int cptr= 0x00;	//ge	eneral purpose counter
. [133]			
	PERIPHEN1 = 0x08;	//E	Enable UARTO
() 016E	mov _PERI	PHEN1,#0x08	
E [135]	PERIPHEN2 = $0x08$;	//E	Enable IO Ports
0 0171	mov _PERI	PHEN2,#0x08	
⊡ □ [136]	P4PINCFG = 0x00;	//Co	onfigure P4 as output
😐 0174	mov _P4PIN	ICFG,#OxOO	
0 [137]			
0 [138]			
E [139]	uartOconfig();	11	/Configure Vart0
0 0177	lcall _ua	rtOconfig	
П. П. Г1407			

Figure 11: Set breakpoint by double clicking on a code line

Once a breakpoint is set, click on the Run button. The processor will then start executing code at full speed until the breakpoint is reached.

[□	PERIPHEN1 = 0x08; mov _PER: PERIPHEN2 = 0x08;	IPHEN1,#0x08	//Enable UARTO	s T				
Ľ	(
×	Name		Scope	Туре	Memory	Size	Address	Value	_
	cptr		INT0Interrupt:B1:L1	char	IRAM	1	0x02	"\0'	
	optr		INT1Interrupt:B1:L1	char	IBAM	1	0x02	"\0'	
	genvar		UART0Interrupt:B1:	char	IRAM	1	0x03	"\0'	
	cptr		main:B1:L1	int	IRAM	2	0x02	0	
	×		delay:B1:L1	BYTE	IRAM	1	0x04	0x00	
힟	dlaisloop		delay:B1:L1	UINT	IBAM	2	0x02	0	
Š	varcheck		Global:B0:L0	char	XRAM	1	0x0010	"\0"	-
Bre								0v016E Line 134 0ddre	ss 0∨016E

Figure 12: Breakpoint reached

Once a breakpoint is reached, you can either:

- Restart the program by clicking again on the Run D button.
- Single step through the code by clicking on the Step D button.
- View/edit SFR, IRAM, XRAM memory locations by clicking on SFR 🗵, IRAM 🎞, XRAM 🕅 buttons.

When the Step D button is clicked, the processor will execute the current instruction and step to the next instruction (and actually stop).

SFR, IRAM, XRAM or FRAM memory content can be viewed and edited anytime a breakpoint is reached. This includes in step mode. The debugger provides table and list view options for the SFR, IRAM, XRAM and FRAM memory areas. It is possible to open more than one window of each one of these memory areas.

Editing a given SFR, IRAM, XRAM and FRAM memory cell is simple:

- Double click on the memory location to edit.
- Type-in the new value to write.
- Press <Enter> or double click on another memory location.

Notes about the SFR edition:

In the case of the SFR memory location, it is important to note that some registers are partially or totally read or write only or can be accessed only when the related peripheral is activated. The SFR edition feature allows the user to control a peripheral up to a certain extent.

6.3 Updating the Code without exiting the debugger

If during a debug session, you find issue in your code and want to modify it, recompile and reload the code into the Flash memory. The Versa Ware JTAG allows you to reload the code in the Flash memory without exiting the Versa Ware JTAG Debugger environment. To do this, simply go to your editor, modify the code, recompile it, return to Versa Ware

Debugger, and click on the Erase then Program $\overline{\bullet}$ button. Note that all the breakpoints you set will be lost.

6.4 Exiting the Debugger

To exit the debugger, click on the Stop Debugging A button. This will return you to the Versa Ware JTAG programming environment.

Disclaimers

Right to make change - Ramtron reserves the right to make changes to its products - including circuitry, software and services - without notice at any time. Customers should obtain the most current and relevant information before placing orders.

Use in applications - Ramtron assumes no responsibility or liability for the use of any of its products, and conveys no license or title under any patent, copyright or mask work right to these products and makes no representations or warranties that these products are free from patent, copyright or mask work right infringement unless otherwise specified. Customers are responsible for product design and applications using Ramtron parts. Ramtron assumes no liability for applications assistance or customer product design.

Life support – Ramtron products are not designed for use in life support systems or devices. Ramtron customers using or selling Ramtron products for use in such applications do so at their own risk and agree to fully indemnify Ramtron for any damages resulting from such applications.

I²C is a trademark of Koninklijke Philips Electronics NV.

Mouser Electronics

Related Product Links

877-VERSAKIT-30XX - Ramtron International VERSAKIT-30XX