



32-SDRAM TLA Logic Analyzer Instruction Manual

Support for ARM Microprocessors
With 32-Bit SDRAM

1 Support Description

The 32-SDRAM Support Package provides timing analysis, state analysis, and both ARM and Thumb disassembly support for ARM microprocessor cores. This package is designed to capture and analyze 32-bit SDRAM bus activity of a single ARM processor.

The 32-SDRAM Support Package requires a Tektronix logic analyzer that has 68 channels or more and runs at least version 3.1 of the TLA software.

2 Support Features

The following summarizes the main features of the 32-SDRAM implementation:

SDRAM CONNECTIVITY

- 16-bit Address Bus
- 32-bit Data Bus
- 4 byte-lane enables
- 2 bank select bits

BUS CYCLE PROTOCOLS

- CAS latency 2 or 3
- Currently, only burst lengths of 1 are supported
- Captures all relevant cycles including: read/write data, instruction fetch, open row, close row, burst terminate, and refresh cycles

DISASSEMBLY

- ARM and Thumb Disassembly
- Little and Big Endian Modes
- User-Specified Base Address Offset (for correct address display)
- Configurable row/column widths
- Supports addressing with up to four open rows

- New display filter mode to filter out all non-instruction/data cycles
- User-Specified Address Ranges For Default Thumb Disassembly
- Intelligent Stream Analysis For Data/Instruction Fetch Identification

3 Installing the 32-SDRAM TLA Support Package

If you received the 32-SDRAM bus support via e-mail, then unpack the 32-SDRAM.zip archive into a temporary folder (either on the TLA itself or accessible over a network by the TLA) and run the Setup.exe from the TLA. If you received a TI_SDRAM installation disk, insert the disk into the TLA floppy disk drive and run the Setup.exe from there.

The installation program will automatically install the package software into the appropriate folders in the file system. For the 32-SDRAM package, the support files will be installed into the “C:/Program Files/TLA700/Supports/32-SDRAM” folder.

4 Loading the 32-SDRAM Support Package

To load the 32-SDRAM support package you must first select the logic analysis module that will be connected to the bus under test. Only TLA 7XX mainframes with multiple modules will have more than one option. To do this, bring up the System window from the Window menu. Click in the title field to select the appropriate module.

With the System window on top, select “Load Support Package...” from the File menu. A list box with all of the installed packages will pop up on the screen, allowing you to pick the package you wish to load into the logic analysis module. Note that the list of support packages shown will depend on which packages have been previously installed on the TLA.

Select “32-SDRAM” from the list and click on the Load button. A dialog box with the following text will appear on the screen:

“Loading a support package invalidates a module’s acquired data. Do you wish to save the current module’s settings and data before loading the support package?”

If you choose “Yes”, a “Save As...” file dialog box will appear, allowing you to save your current module setup and data. Selecting “No” will cause the TLA to overwrite your current setup and data with the 32-SDRAM package setup.

5 32-SDRAM Signal Requirements

The 32-SDRAM support package uses the following signals:

Clocking:	CLK, CKE
Bank Select	BA0, BA1
Row/Col Address:	A15-A0
Data:	D31-D0
Control	RAS_, CAS_, WE_
Chip Select	CS_
Byte-lane enables	BE3_-BE0_

Signals that end with ‘_’ are active low. All signals are assumed to be at TTL logic levels. Except for unused address signals, all signals must be connected for proper bus support operation. CKE, if unused, should be pulled high.

6 Connecting the TLA to the 32-SDRAM Bus

If your system uses P6434 Mictor¹ connectors, Tables 1 and Table 2 describe the required signal pin-outs.² Note that Agilent and Tektronix use different Mictor pin numberings; both numbering schemes are provided in tables 1 and 2. Unused signals are available for user-defined signal connections.

Note that if you have a TLA with more than 68 channels, you will need to connect your Mictor cables in a NON-STANDARD way, as half of the C/D connector needs to connect to the C3/C2 port of the TLA and half needs to connect to the D1/D0 port. For example, if you use a standard D-labeled Mictor cable, the D3/D2 connector from the cable must go into the C3/C2 port of the TLA and the D1/D0 connector is connected as normal to the D1/D0 port of the TLA. This is necessary for compatibility with the 68-channel TLA connector format.

If your system uses square-pin header connectors, use Table 1 and Table 2 to connect up your TLA via 6417 flying lead-set connectors.

¹ The Mictor connector is described in Tektronix document 070-9793-02, available from Tektronix at www.tek.com.

² If your system uses Mictor connectors but does not adhere to the TI-SDRAM Mictor specification, NEX-HDSWIZ adaptors are available from New Wave PDG. See www.busboards.com for adaptor specifications and ordering information.

Table 1. A CONNECTOR

	Tek	A3/A2	Agilent	A1/A0	Tek		
	1	NC	1	2	NC	38	
	2	NC	3	4	NC	37	
unused	3	CLK:0	5	6	CLK:1	36	
D31	4	A3:7	7	8	A1:7	35	D15
D30	5	A3:6	9	10	A1:6	34	D14
D29	6	A3:5	11	12	A1:5	33	D13
D28	7	A3:4	13	14	A1:4	32	D12
D27	8	A3:3	15	16	A1:3	31	D11
D26	9	A3:2	17	18	A1:2	30	D10
D25	10	A3:1	19	20	A1:1	29	D9
D24	11	A3:0	21	22	A1:0	28	D8
D23	12	A2:7	23	24	A0:7	27	D7
D22	13	A2:6	25	26	A0:6	26	D6
D21	14	A2:5	27	28	A0:5	25	D5
D20	15	A2:4	29	30	A0:4	24	D4
D19	16	A2:3	31	32	A0:3	23	D3
D18	17	A2:2	33	34	A0:2	22	D2
D17	18	A2:1	35	36	A0:1	21	D1
D16	19	A2:0	37	38	A0:0	20	D0

Table 2. C/D CONNECTOR

	Tek	C3/C2	Agilent	D1/D0	Tek		
	1	NC	1	2	NC	38	
	2	NC	3	4	NC	37	
CLK	3	CLK:3	5	6	CLK:2	36	CKE
unused	4	C3:7	7	8	D1:7	35	A15
BE3_	5	C3:6	9	10	D1:6	34	A14
unused	6	C3:5	11	12	D1:5	33	A13
unused	7	C3:4	13	14	D1:4	32	A12
unused	8	C3:3	15	16	D1:3	31	A11
BE2_	9	C3:2	17	18	D1:2	30	A10
unused	10	C3:1	19	20	D1:1	29	A9
BE1_	11	C3:0	21	22	D1:0	28	A8
BE0_	12	C2:7	23	24	D0:7	27	A7
unused	13	C2:6	25	26	D0:6	26	A6
BA1	14	C2:5	27	28	D0:5	25	A5
BA0	15	C2:4	29	30	D0:4	24	A4
WE_	16	C2:3	31	32	D0:3	23	A3
RAS_	17	C2:2	33	34	D0:2	22	A2
CAS_	18	C2:1	35	36	D0:1	21	A1
CS_	19	C2:0	37	38	D0:0	20	A0

7 Custom Clocking Options

The 32-SDRAM support package provides two main clocking modes: “Valid SDRAM Cycles” and “Rising Edge of Clk (debug)”.

7.1 Valid SDRAM Cycles

This is the default clocking mode, and configures the 32-SDRAM package to acquire records for valid SDRAM bus cycles. All ‘interesting’ cycles such as open row, precharge, refresh, burst terminate, and CAS read/write cycles are acquired in this mode. Also, due to the requirements of the TLA clocking state machine, latency cycles between CAS read and data valid are also acquired (and labeled ‘SDRAM-PIPELINE’).

The remaining clocking options are applicable in this mode. They are the ‘CAS Latency’ and ‘Burst Length’.

7.1.1 CAS Latency

An SDRAM CAS latency of 2 or 3 is supported. This specifies the number of clock cycles after CAS that read-data is valid.

7.1.2 Burst Length

Currently, disassembly is supported only for burst lengths of 1. However, the clocking option has been provided to select burst lengths between 1 and 4. While untested, burst lengths greater than 1 should at least capture the correct raw bus records.

7.2 Rising Edge of Clk (debug)

This clocking mode is used only for low-level bus cycle debugging purposes and will not result in valid disassembly. When this mode is selected, none of the other clocking options have any effect, and all signals will be captured unconditionally on the rising edge of Clk.

8 Symbol Tables

Two symbol tables are provided in this support package. To set the display mode for a group, right-click on the group column header in the Listing window and select Properties... Use the radix drop box to choose the display mode. To display a group using a symbol table, select the ‘symbolic’ menu option and specify the symbol file to use.

- The 32-SDRAM_attr.tsf symbol file is used by the -display attribute-group. It sets up the colors used for highlighting sequences based on cycle type.
- The 32-SDRAM_mctrl.tsf symbol file is used by the MemCtrl group. It interprets the cycle type of a sequence.

Symbol Table for the –display attribute- Group

<i>Symbol</i>	<i>Group</i>	<i>Text/Background</i>
SPECIAL	1111	@white/@gray
SUBROUTINE	0101	@black/@yellow
CONTROL FLOW	0100	@black/@cyan
INSTRUCTION	0011	@black/@silver
INST_DATA	0010	@black/@white
HARDWARE	0001	@black/@white
UNKNOWN	XXXX	@white/@gray

Symbol Table for the MemCtrl Group

<i>Symbol</i>	<i>Group</i>	<i>Description</i>
LOAD REG	0000	Load mode register
REFRESH	0001	Refresh cycle
CLOSE ROW	0010	Precharge
OPEN ROW	0011	Open row
WRITE	0100	CAS write cycle
READ	0101	CAS read cycle
BURST_TERM	0110	Burst terminate
NOP	0111	No operation
IDLE	1111	Idle bus
OTHER	1XXX	Other bus activity

9 Disassembly

Once the 32-SDRAM support package has been loaded and a trace taken, the Listing display for the logic analysis module will automatically display disassembled data. Two methods are available to the user to configure and control disassembly: user marks and user options. User marks allow a bus cycle to be tagged, for example as an ARM or Thumb instruction or a read data access. These marks are used as reference points for the disassembler when there is ambiguity in the data stream. User options are numeric fields and drop-down selectable lists to configure or format disassembly.

9.1 Marking Mnemonics

To mark an element in the Listing window, right-click on the mnemonic and select ‘Mark Opcode’ from the menu. This will bring up a menu of marking options, including the option to remove an existing mark. Specific marking options include:

- ARM Opcode
- Thumb Opcode
- Read Data Access
- Undo
- Debug Mark

Marking an element forces the disassembler to decode according to the specified cycle type. This is sometimes necessary to obtain correct disassembly, usually at the beginning of a trace where there is not enough context to fully disambiguate cycle types.

Use the ‘ARM Opcode’ and ‘Thumb Opcode’ marks to indicate an ARM or Thumb instruction fetch respectively. Use the ‘Read Data Access’ mark to indicate a data fetch. The ‘No branch’ mark can be used to force disassembly of the flush pipeline following a branch instruction as if the branch was not taken (whether or not this is true). This allows the two instructions in the branch pipeline to be viewed as if they were executed rather than marked as flushed. Use the ‘Undo’ mark to remove a mark on an element. Finally, the ‘Debug Mark’ is useful to Dragonfly only, causing the display of internal state variables from the disassembler.

9.2 Disassembly Options

Disassembly options can be found under the Disassembly tab of the Listing display properties pages. To get to these pages, right-click in the Listing data window and select ‘Properties’ from the pop-up menu. The following disassembly options are available for the 32-SDRAM support package (see the next section for more details on how the disassembler uses many of these options).

- **Reg Names** Selects the disassembly register format, either ‘Symbolic’ or ‘Rnumber’. Symbolic register names are as follows: PC = R15, LR = R14, SP = R13, IP = R12, FP = R11.
- **Endianess** Selects the endian mode of the bus, either ‘Little’ or ‘Big’.
- **Burst Length** Configures the SDRAM burst length. Currently, only burst length of 1 is supported for disassembly. Note that if you change this value, you **MUST** also change the burst length in the custom clocking options for correct trace capture. Unfortunately, the Tektronix TLA does not provide for easy communication between the disassembler and clocking state capture logic.
- **CASlatency** Specifies the SDRAM CAS latency – the number of cycles delay between CAS read assertion and valid data. Note that if you change this value, you **MUST** also change the latency in the custom clocking options for correct trace capture.
- **CASBits** Specifies the number of valid column bits (not including A10, which is used for auto-precharge indication).
- **RASBits** Specifies the number of valid row bits (not including the bank bits).
- **BankBits** Number of bank bits – only 2 are supported in the 32-SDRAM package.
- **SDRAMBaseAddr** Hexadecimal base offset to add to disassembly display. This offset is only for display purposes and cannot be used in the complicated procedure for SDRAM address triggering.
- **ROMBaseAddr** Currently unused.

- **DBWidth** Specifies 16-bit or 32-bit data bus. Currently, only 32-bit data bus is supported.
- **Thumb Address Range Start1** Denotes the starting address of a range to associate with default Thumb disassembly.
- **Thumb Address Range End1** Denotes the ending address + 1 of a range to associate with default Thumb disassembly.

9.3 Disassembling SDRAM Bus Cycles

This section describes how the 32-SDRAM package disassembles bus cycles. It is not the intent here to document the SDRAM bus protocol, only behavior as it relates to disassembly. Please refer to your SDRAM memory data sheet for details of operation.

The SDRAM bus protocol is complicated in that the complete address of a data element must be derived from two sequences. The column address is obtained from the associated CAS sequence, which is predicatably located depending on cycle-type (read or write), burst length, and position in burst. Also contained in the CAS sequence are the bank bits, which must be matched to find the correct prior RAS sequence that will yield the row address of the data element. However, the RAS sequence is not predictably located, and long searches in the trace capture may have to be made to locate it.

More generally, the SDRAM bus protocol does not provide any ARM specific cycle type information. That is, there is no explicit indication of ARM fetch versus Thumb fetch versus data fetch. However, the byte-lane enables do allow the disassembler to distinguish ARM versus Thumb, but not ARM fetch versus data reads, or Thumb fetch versus 16-bit data reads.

One feature the 32-SDRAM support package is its capability to do sophisticated stream analysis to determine the cycle type. For example, if a load instruction is executed, the disassembler can try to find and associate one or more cycles subsequent in the stream and mark them as data reads.

However, several factors complicate this method. A load/store instruction can be conditionally executed or its address could reference internal memory (if present on the core), and thus the read/write data transaction is never seen on the bus. Also, pipeline delays in the execution unit cause intervening instruction fetches to be seen on the bus before data reads/writes from load/store instructions are seen.

Accordingly, the disassembler attempts to make intelligent inferences not only from disassembled opcodes, but also from discontinuities in the address stream, user marks, and user specified default Thumb address ranges. Users can mark cycles as ARM or Thumb instructions, or data accesses. Often a single mark can correct an entire disassembly listing. This is because of the dependencies of future cycles on the current cycle type. For example, the user can mark a Thumb instruction (in an address range not defaulted for Thumb) and the disassembler will then naturally infer that all subsequent instructions must be Thumb instructions until a BX instruction is encountered.

As the 32-SDRAM package was derived from a more general bus support, some of its features will likely not be required. For example, ARM versus Thumb disassembly should never be incorrect, assuming byte-lane enables are connected and working properly. So, the default Thumb Start and End address range should not be necessary.

The disassembly display can be filtered to show only certain cycle types. The cycle-type classification is a hierarchy where each subsequent cycle-type is a subset of the preceding one. Five cycle-type classifications are defined for this bus support.

- **HARDWARE** All sequences
- **INST_DATA** Instruction fetches, data reads/writes
- **INSTRUCTION** Instruction fetches
- **FLOW_CONTROL** Branch instructions taken
- **SUBROUTINE** Branch and link instructions

In addition to the standard cycle-types, the 32-SDRAM bus support defines the **INST_DATA** cycle-type, which excludes all sequences except for instruction fetches, data reads and data writes. To select cycle-type filtering, right-click in the Listing window, choose Properties, then select the Disassembly tab. Use the **SHOW** drop-down list to select the level of cycle-type display filtering.

9.4 Address Display and Triggering

In the 32-SDRAM package, two address groups can be displayed in the Listing window: ‘Address’ and ‘32-SDRAM Addr’. The ‘Address’ field displays the raw bus data that was on the address bus during that clock cycle. The ‘32-SDRAM Addr’ displays the derived address associated with the current sequence, and by default is the only visible address group. Columns can be added by right-clicking in the Listing window and selecting Add Column.

The derived address is built from the following:

- SDRAM base offset specified by user (Disassembly option)
- Bank address from bank bits
- Row address from RAS sequence that matches bank bits
- Column address from CAS sequence
- Burst position
- Data bus width
- Byte offset for 16-bit or 8-bit data reads/writes (according to endian ordering)

When neither the row or column address can be found for a given sequence (generally at the beginning of a trace), it will be labeled: **UNKNOWN CYCLE**. If a row address cannot be found but a column address can be found, the address will have only 6 valid digits with the upper two digits displayed as “—“.

Triggering on SDRAM addresses can be problematic. A triggering state machine must be built based on the raw address, which cannot account for most of the above components comprised by the cooked address. Moreover, the unused upper bits in the raw address must be marked as don't cares in the trigger state machine since they are not, in general, driven to ground during RAS and CAS cycles.

The general format for such a triggering state machine is the following:

- 1) Search for Address = <row addr> && bank = <bank addr> && !RAS_ &&
!CS_ && CAS_ goto state 2
- 2) Search for Address = <cas addr> && bank = <bank addr> && RAS_ && !CS_
&& !CAS_ trigger

10 DRAGONFLY SOFTWARE LICENSE AGREEMENT

PLEASE READ THIS DOCUMENT CAREFULLY BEFORE USING THE SOFTWARE. BY USING THE SOFTWARE, YOU ARE AGREEING TO BE BOUND BY THE TERMS OF THIS AGREEMENT. IF YOU DO NOT AGREE TO THE TERMS OF THIS AGREEMENT, DO NOT OPEN THE SEALED DISK PACKAGE, INSTALL, OR USE THE SOFTWARE. PROMPTLY RETURN WITHIN 15 DAYS THE SOFTWARE, ALL RELATED DOCUMENTATION, AND ACCOMPANYING ITEMS TO THE PLACE OF ACQUISITION FOR A FULL REFUND.

This is a legal agreement between you and Dragonfly Software Development (Dragonfly). This Agreement states the terms and conditions upon which Dragonfly offers to license the software sealed in the disk package, together with all related documentation and accompanying items including but not limited to, the executable programs, drivers, libraries, and data files associated with such programs (collectively, the Software).

LICENSE

1. Grant of License

The Software is licensed, not sold, to you for use only under the terms of this Agreement. You own the disk or other media on which the Software is originally or subsequently recorded or fixed as permitted by this Agreement. However, as between you and Dragonfly (and, to the extent applicable, its licensors), Dragonfly retains all right, title, and interest to the Software and all copyrights to the Software, and reserves all rights not expressly granted to you. This is a non-exclusive license.

2. For Use on a Single Computer

The Software may be used by you only on a single computer. You may transfer the machine-readable portion of the Software from one computer to another computer, provided that (a) the Software (including any portion or copy thereof) is erased from the first computer and (b) there is no possibility the Software will be used on more than one computer at a time.

3. One Archival Copy

In support of your use of the Software on a single computer, you may make one (1) archival copy of the machine-readable portion of the Software for back up purposes only, provided that you reproduce on the copy all copyright and other proprietary rights notices included on the originals of the Software.

4. Transfer of License

You may transfer your license of the Software, provided that (a) you transfer all portions of the Software or copies thereof, (b) you do not retain any portion of the Software or any copy thereof, and (c) the transferee reads and agrees to be bound by the terms and conditions of this Agreement.

5. Limitation on Using, Copying, and Modifying the Software

Except to the extent expressly permitted by this Agreement or by the laws of the jurisdiction where you acquired the Software, you may not use, copy, or modify the Software. Nor may you sub-license any of your rights under this Agreement.

6. Decompiling, Disassembling, or Reverse Engineering

You acknowledge that the Software contains trade secrets and other proprietary information of Dragonfly and its licensors. Except to the extent expressly permitted by this Agreement or by the laws of the jurisdiction where you are located, you may not decompile, disassemble, or otherwise reverse engineer the Software, or engage in any other activities to obtain underlying information that is not visible to the user in connection with normal use of the Software. In any event, you will notify Dragonfly of any information derived from reverse engineering or such other activities, and the results thereof will constitute the confidential information of Dragonfly that may be used only in connection with the Software.

TERMINATION

The license granted to you is effective until terminated. You may terminate the license at any time by returning the Software (including any portions or copies thereof) to Dragonfly at the address shown below. The license will also terminate automatically without any notice from Dragonfly, if you fail to comply with any term or condition of this Agreement. You agree upon such termination to return the Software (including any portions or copies thereof) to Dragonfly. Upon termination, Dragonfly may also enforce any rights provided by law. The provisions of this Agreement that protect the proprietary rights of Dragonfly will continue in force after termination.

LIMITED WARRANTY

Dragonfly warrants, as the sole and exclusive warranty, that the disks on which the Software is furnished will be free of defects for a period of ninety (90) days. In the event one or more of such disks is defective, Dragonfly will replace the defective disk(s) free of charge upon receiving the defective disk at the address set forth below.

No distributor, dealer, or any other entity or person is authorized to expand or alter this warranty or any other provisions of this Agreement. Any representation, other than this express limited warranty, will not bind Dragonfly.

EXCEPT AS STATED ABOVE IN THIS AGREEMENT, THE SOFTWARE IS PROVIDED AS-IS WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Dragonfly does not warrant that the functions contained in the Software will meet your requirements, or that the operation of the Software will be uninterrupted or error-free. You assume full responsibility for the selection of the Software to achieve your intended results, and for the installation, use, and results obtained from the Software. You also assume the entire risk as it applies to the quality and performance of the Software. Should the Software prove defective you (and not Dragonfly, or its distributors or dealers) assume the entire cost of all necessary servicing, repair, or correction.

This warranty gives you specific legal rights, and you may also have other rights which vary from country/state to country/state. Some countries/states do not allow the exclusion of implied warranties, so the above exclusion may not apply to you. Dragonfly disclaims all warranties of any kind of the Software was customized, repackaged, or altered in any way by any third party other than Dragonfly.

LIMITATION OF REMEDIES AND DAMAGES

THE ONLY REMEDY FOR BREACH OF WARRANTY IS THE EXPRESS LIMITED WARRANTY SET FORTH ABOVE. IN NO EVENT WILL DRAGONFLY OR ITS LICENSORS BE LIABLE FOR ANY PUNITIVE, INDIRECT, INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES, OR FOR ANY LOST PROFITS, LOST SAVINGS, LOST REVENUES, OR LOST DATA ARISING FROM OR RELATING TO THE SOFTWARE OR THIS AGREEMENT, EVEN IF DRAGONFLY OR ITS LICENSORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN NO EVENT WILL DRAGONFLY'S LIABILITY OR DAMAGES TO YOU OR ANY OTHER PERSON EVER EXCEED THE AMOUNT PAID BY YOU TO USE THE SOFTWARE, REGARDLESS OF THE FORM OF THE CLAIM. Some countries/states do not allow the limitation or exclusion of liability for the incidental or consequential damages, so the above limitation or exclusion may not apply to you.

PRODUCT RETURNS

If you must ship the Software to Dragonfly or an authorized Dragonfly distributor or dealer, you must prepay shipping and either insure the software or assume all risk of loss or damage in transit.

U.S. GOVERNMENT RESTRICTED RIGHTS

All Software and related documentation are provided with restricted rights. Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 or the Commercial Computer Software-Restricted Rights at 48 CFR 52.227-19, as

applicable. If you are sub-licensing or using the Software outside of the United States, you will comply with the applicable local laws of your country, U.S. export control law, and the English version of this Agreement.

MANUFACTURER

Dragonfly Software Development
4905 SW Griffith Drive #100
Beaverton OR 87005
(503)-643-3800

GENERAL

This Agreement is binding on you as well as your employees, employers, contractors and agents, and on any successors and assignees. Neither the Software nor any information derived therefrom may be exported except in accordance with the laws of the U.S. or other applicable provisions. This Agreement is governed by the laws of the State of Oregon (except to the extent federal law governs copyrights and federally registered trademarks). This Agreement is the entire agreement between us and supersedes any other understandings or agreements, including but not limited to, advertising of the Software. If any provision of this Agreement is deemed invalid or unenforceable by any country or government agency having jurisdiction, that particular provision will be deemed modified to the extent necessary to make the provision valid and enforceable, and the remaining provisions will remain in full force and effect. If any legal action is brought by you or Dragonfly regarding the Software or this Agreement, the prevailing party shall be entitled to recover, in addition to any other relief granted, reasonable attorney fees and expenses of litigation. Neither you nor Dragonfly will waive any rights under this Agreement, unless such waiver is in writing. For questions concerning the Software or this Agreement, please contact Dragonfly at the address stated above.