



# ARM-EBI TLA Logic Analyzer Instruction Manual

Support for ARM Microprocessors  
With the Atmel EBI Bus

## 1 Support Description

The ARM-EBI 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 the Atmel EBI bus activity of a single ARM processor.

Since the ARM is a microprocessor core, it is not possible to provide a support package that will work correctly with all possible ARM configurations, due to specific ASIC implementation differences that might arise. However, this support is compatible with the Atmel EBI bus specification, and is designed to operate directly on Atmel's AT91EB01 Evaluation Board, as well as other boards that provide access to the required address, data and bus control signals. The AT91 data sheet which contains the EBI specification can be found at [www.atmel.com](http://www.atmel.com): search for EBI and download the AT91M40400 (complete) manual.

The ARM-EBI 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 ARM-EBI implementation:

### EBI CONNECTIVITY

- 24-bit Address Bus
- 16-bit Data Bus
- 3 Chip Selects (NCS0, NCS1, NCS2)
- External Wait (NWAIT)

### BUS CYCLE PROTOCOLS

- Standard-Read and Early-Read Protocols
- Byte-Write Access Mode (NRD, NWR0, NWR1 signals)
- Byte-Select Access Mode (NOE, NWE, NUB, NLB signals)
- Independently Configurable Internal Wait States (up to 6 cycles)

### DISASSEMBLY

- ARM and Thumb Disassembly
- Little and Big Endian Modes
- User-Specified Chip Select Base Addresses (for correct address display)
- Address Bus Width Specification (for correct address display <24 bits)
- User-Specified Address Ranges For Default Thumb Disassembly
- Intelligent Stream Analysis For Data/Instruction Fetch Identification

### UNSUPPORTED

- 8-bit data bus
- Multiplexed Chip Select/Address lines and NCS3
- Multiple processor systems

### **3 Installing the ARM-EBI TLA Support Package**

If you received the ARM-EBI bus support via e-mail, then unpack the ARM-EBI.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 an ARM-EBI 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 ARM-EBI package, the support files will be installed into the “C:/Program Files/TLA700/Supports/ARM-EBI” folder.

## 4 Loading the ARM-EBI Support Package

To load the ARM-EBI 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 “ARM-EBI” 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 ARM-EBI package setup.

## 5 ARM-EBI Signal Requirements

The ARM-EBI support package uses the following signals:

Clock:	MCKI
Address:	A23-A1
Data:	D15-D0
Control	NRD/NOE, NWAIT, NWR0/NWE, NWR1/NUB, A0/NLB
Chip Select	NCS2, NCS1, NCS0

Signals that begin with the letter ‘N’ are active low. All signals are assumed to be at TTL logic levels. All data bits and clock are required. Chip select signals from each memory bank/type must be connected to any or NCS2, NCS1, or NCS0; those unused may be left unconnected.

The EBI specification defines two read protocols: early-read and standard-read. Early-read allows NRD/NOE to go low sooner in the first clock of a read bus cycle. As a result, back-to-back read cycles may only be implicitly inferred and decoded from knowledge of the internal wait-state cycles (see the Custom Clocking Options section for details on how to configure the internal wait states for the ARM-EBI package). Thus, if your system uses the standard-read protocol, or does not use the external wait signal, NWAIT is not required.

The rest of the control signals required by the ARM-EBI package depend on which write access mode is used in your system. The EBI specification defines two modes: byte-write and byte-select. In byte-write mode, the control signals NRD, NWR0, and NWR1 are required. In byte-select mode, the control signals NOE, NWE, NUB, and NLB are required. Which protocol is used must be selected for proper decoding of bus cycles.<sup>1</sup>

Only the address bus signals used in your system are required by the ARM-EBI support package. The width of the address bus is a user configurable disassembly option, allowing bus widths up to 24-bits. With this option, unused address signals or those that have been assigned a PIO function (a configurable option in the EBI specification) can be masked for disassembly and display.<sup>2</sup>

---

<sup>1</sup> See the *Custom Clocking Options* section for more details.

<sup>2</sup> See the *Disassembly* section for more details.

## 6 Connecting the TLA to the ARM-EBI Bus

If your system uses P6434 Mictor<sup>3</sup> connectors, Tables 1 and Table 2 describe the required signal pin-outs.<sup>4</sup> 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 A0 from the ARM processor is usually connected to the chip select or byte-lane select of external memory (or used in the logic that provides these signals). And so, A1 from the processor is connected to A0 of external memory. A common mistake is to then connect A1 from the processor to A0 on the mictor connector (and A2 to A1, A3 to A2, and so on). This will make you unhappy for three reasons: 1) addresses displayed in the disassembly will be halved from the correct addresses, 2) the byte-lane indication in the disassembly for data writes will be incorrect and 3) triggering the TLA will require you to enter the desired address divided by 2. Please connect A0 to A0, A1 to A1, etc. from the processor to the mictor.

Unused upper address bits on the mictor connector should be grounded. If they are left unconnected or even if other user signals are connected to these inputs, it is possible to configure the bus support package to disassemble correctly. Please see section 9.4 on address display for more details.

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.

---

<sup>3</sup> The Mictor connector is described in Tektronix document 070-9793-02, available from Tektronix at [www.tek.com](http://www.tek.com).

<sup>4</sup> If your system uses Mictor connectors but does not adhere to the ARM-EBI Mictor specification, NEX-HDSWIZ adaptors are available from New Wave PDG. See [www.busboards.com](http://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
MCKI	3	CLK:0	5	6	CLK:1	36
unused	4	A3:7	7	8	A1:7	35
unused	5	A3:6	9	10	A1:6	34
unused	6	A3:5	11	12	A1:5	33
unused	7	A3:4	13	14	A1:4	32
unused	8	A3:3	15	16	A1:3	31
unused	9	A3:2	17	18	A1:2	30
unused	10	A3:1	19	20	A1:1	29
unused	11	A3:0	21	22	A1:0	28
A23	12	A2:7	23	24	A0:7	27
A22	13	A2:6	25	26	A0:6	26
A21	14	A2:5	27	28	A0:5	25
A20	15	A2:4	29	30	A0:4	24
A19	16	A2:3	31	32	A0:3	23
A18	17	A2:2	33	34	A0:2	22
A17	18	A2:1	35	36	A0:1	21
A16	19	A2:0	37	38	A0:0	20

NRD/NOE  
A15  
A14  
A13  
A12  
A11  
A10  
A9  
A8  
A7  
A6  
A5  
A4  
A3  
A2  
A1  
A0/NLB

**Table 2. C/D CONNECTOR**

	Tek	C3/C2	Agilent	D1/D0	Tek	
	1	NC	1	2	NC	38
	2	NC	3	4	NC	37
NWAIT	3	CLK:3	5	6	CLK:2	36
unused	4	C3:7	7	8	D1:7	35
unused	5	C3:6	9	10	D1:6	34
unused	6	C3:5	11	12	D1:5	33
unused	7	C3:4	13	14	D1:4	32
unused	8	C3:3	15	16	D1:3	31
unused	9	C3:2	17	18	D1:2	30
unused	10	C3:1	19	20	D1:1	29
unused	11	C3:0	21	22	D1:0	28
unused	12	C2:7	23	24	D0:7	27
unused	13	C2:6	25	26	D0:6	26
unused	14	C2:5	27	28	D0:5	25
unused	15	C2:4	29	30	D0:4	24
NWR1/NUB	16	C2:3	31	32	D0:3	23
NCS2	17	C2:2	33	34	D0:2	22
NCS1	18	C2:1	35	36	D0:1	21
NCS0	19	C2:0	37	38	D0:0	20

NWR0/NWR  
D15  
D14  
D13  
D12  
D11  
D10  
D9  
D8  
D7  
D6  
D5  
D4  
D3  
D2  
D1  
D0

## 7 Custom Clocking Options

The ARM-EBI support package provides two main clocking modes: “Valid EBI Cycles” and “Both Edges of MCKI”.

### 7.1 *Valid EBI Cycles*

This is the default clocking mode, and configures the ARM-EBI package to acquire records for valid EBI bus cycles. The remaining clocking options are applicable in this mode. They are the ‘EBI Read Protocol’ and ‘Chip Select Wait States’ for NCS0, NCS1, and NCS2.

#### 7.1.1 EBI Read Protocol

As mentioned previously, the EBI read protocol can be configured as early-read or standard-read. The early-read protocol is the default. This mode requires appropriate settings of the internal wait-states for each chip select that is used, as well as valid signaling on the NWAIT line (if used).

The standard-read protocol does NOT use the internal wait-state settings or the external NWAIT signal. Instead, it monitors the read/write signals going inactive to latch in valid signals. Thus, ARM wait states, whether internal or external, are implicitly determined by the active duration of the read/write signals. As a result, it might be possible in some implementations to use the standard-read clocking option (and not have to configure wait states), but only if the implementation inserts at least one idle clock cycle in between read accesses. Otherwise early-read must be used, and wait-states must be specified.

#### 7.1.2 Chip Select Wait States

These clocking options configure each of the three chip selects NCS0, NCS1, and NCS2 for the correct number of internal wait-states. The default for each is “0 wait states”. Options allow selection of 0-6 wait states or disabled. Chip selects only need to be disabled if they are active, but should not be traced (e.g. SDRAM chip selects or if NCS2 is configured as PIO).

### 7.2 *Both Edges of MCKI*

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 both the rising and falling edge of MCKI.

## 8 Symbol Tables

Three symbol tables are provided in this support package. Only the symbol table for the chip select group is displayed by default. The control group symbol table is optionally selected by the user, based on the write access mode used in the system under test (byte-select or byte-write).

- The ARM-EBI\_cs.tsf symbol file is used by the chip select group.
- The ARM-EBI\_Ctrl\_BS symbol file is used by the control group when byte-select access mode is used.
- The ARM-EBI\_Ctrl\_BW symbol file is used by the control group when byte-write access mode is used.

*Symbol Table for the Chip Select Group*

<i>Symbol</i>	<i>Group</i>	<i>Description</i>
CHIP_SEL_2	011	Chip select 2 asserted
CHIP_SEL_1	101	Chip select 1 asserted
CHIP_SEL_0	110	Chip select 0 asserted
IDLE	111	No chip select
Undefined	XXX	Conflicting chip selects

Group Signals are NCS2, NCS1, NCS0.

*Symbol Table for the Control Group (byte-select access)*

<i>Symbol</i>	<i>Group</i>	<i>Description</i>
READ	01XX	Read cycle
WRITE_16	1000	16-bit write
WRITE_U8	1001	Upper-byte write
WRITE_L8	1010	Lower-byte write
IDLE	11XX	No read/write
Undefined	XXXX	Conflicting read/write

Group Signals are NOE, NWR, NUB, NLB.

*Symbol Table for the Control Group (byte-write access)*

<i>Symbol</i>	<i>Group</i>	<i>Description</i>
READ	011X	Read cycle
WRITE_16	100X	16-bit write
WRITE_U8	110X	Upper-byte write
WRITE_L8	101X	Lower-byte write
IDLE	111X	No read/write
Undefined	XXXX	Conflicting read/write

Group Signals are NRD, NWR0, NWR1, A0

## 9 Disassembly

Once the ARM-EBI 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 as an ARM or Thumb instruction, or a 32, 16 or 8-bit 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 may include:

- ARM Opcode
- Thumb Opcode
- 32-bit Data Access
- 16-bit Data Access
- 8-bit Data Access
- No branch
- 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 '32-bit Data Access', '16-bit Data Access', and '8-bit Data Access' marks to indicate the start of a data read/write. When a 32-bit cycle type is specified (i.e. ARM opcode or 32-bit data), the disassembler will correctly denote the subsequent extension cycle that completes the data element (as ARM extension or Data extension respectively).

The 'No branch' mark can be used to force disassembly on a branch instruction as if the branch was not taken (whether or not this is true). This prevents instructions in the branch pipeline to be flushed and they will be decoded as if executed. Use the 'Undo' mark to remove a mark on an element. The 'Debug Mark' is useful to Dragonfly only, causing the display of internal state variables from the disassembler.

The marking options available depend on what the address of the element to be marked. If the address is not a multiple of 4 or the address of the subsequent element is not contiguous (i.e. +2) then the options 'ARM Opcode' and '32-bit Data Access' will not be

available. Two, consecutive, aligned, 16-bit accesses are required to build the 32-bit instruction or data implied by these markings.

## 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 ARM-EBI support package (see the next section for more details on how the disassembler uses many of these option).

- **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.
- **Endianness** Selects the endian mode of the bus, either 'Little' or 'Big'.
- **EBI Write Access Mode** Selects the write protocol used by the system 'Byte\_Select' or 'Byte\_Write'.
- **Address Bus Width** Specifies the number of valid address bits (0-24 allowed). The disassembler will mask bits outside the given range to zero.
- **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.
- **Thumb Address Range Start2** Denotes the starting address of a second range to associate with default Thumb disassembly.
- **Thumb Address Range End2** Denotes the ending address + 1 of a second range to associate with default Thumb disassembly.
- **Chip Select 0 Base Address** Specifies the starting address for chip select 0. This address will be added to bus addresses for display.
- **Chip Select 1 Base Address** Specifies the starting address for chip select 1. This address will be added to bus addresses for display.
- **Chip Select 2 Base Address** Specifies the starting address for chip select 2. This address will be added to bus addresses for display.

## 9.3 Disassembling Atmel EBI Bus Cycles

This section describes how the ARM-EBI package supports the Atmel EBI specification. EBI is a generic bus protocol with minimal embedded cycle type information. Only read/write, and byte-lane indicators for writes are available. Consequently, it can be difficult or impossible to determine whether a bus read cycle is fetching a Thumb instruction, part of an ARM instruction, 16-bit data, or part of a 32-bit piece of data without additional information.

One feature the ARM-EBI 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 a data reads.

However, several factors complicate this method. A load/store instruction can be conditionally executed or its address could reference internal memory, and thus the read/write data transaction is never seen on the bus. 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 32-, 16-or 8-bit 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.

Because of the way the TLA software invokes disassembly, unexpected behavior can sometimes occur. Assume for a moment that no default Thumb address ranges have been defined. Then, consider the disassembly of a thumb routine called from ARM code. At first, these instructions will be incorrectly disassembled as ARM instructions. But, when the user marks the first Thumb instruction appropriately, then subsequent disassembly should be correctly interpreted as Thumb until the next BX instruction.

If this BX instruction is a return from the Thumb routine back to ARM code, then the disassembler will continue to parse correctly (since it will use the default Thumb address ranges to decide Thumb/ARM and, in this case, no default Thumb address ranges are defined). Alternately, if the BX branches to more Thumb code, then disassembly will not be correct again and will need another user mark.

However, when the user marks the next instruction as Thumb, the TLA backtracks to re-disassemble the instructions previous to the current one being marked. Unfortunately, it does not backtrack all the way to the beginning of the trace, instead it backtracks only part way back, perhaps not even as far back as the first user mark (at the beginning of the Thumb routine). In this case, disassembly will be incorrect up until the second user mark (the one just made) because the disassembler will not have any ARM/Thumb typing information and will default to ARM.

In such cases, it is best to define a default Thumb address range that includes the Thumb assembly routine of interest. This will also be useful for Thumb routines that get called more than once in a trace, since the user would otherwise need to repeatedly mark the first instruction of the routine. As a general rule, it is recommended that Thumb routines are linked into their own memory region that can be identified to the disassembler with a single Thumb address range.

## 9.4 Address Display and Triggering

In the ARM-EBI package, two address groups can be displayed in the Listing window: 'ARM-EBI Addr' and 'Address'. By default, 'ARM-EBI Addr' is the only visible address group, and contains the following modifications to the 24-bit raw bus address contained in the 'Address' group:

- Unused address bits as indicated by the Address Bus Width field are masked.
- The associated chip select base address is added.
- A0 is masked to obtain 16-bit word aligned addresses.

Note that A0 is masked in the ARM-EBI Addr group display, as it is either unused (byte-write access) or used for the NLB signal (byte-select access).

If upper address bits have been left unconnected, they will float high. If left unmasked by the Address Bus Width field option, they will show as '1's in the address. For correct display, you can use the Address Bus Width option. However, **because disassembly options apply to the display only**, for correct triggering (i.e. triggering the TLA to begin capture at a specific address), you will need to modify the Address group definition in the ARM-EBI Setup window on the TLA. To do this:

- 1) Load the ARM-EBI support package.
- 2) Bring up the Setup window (menu option Window->Setup: ARM-EBI).
- 3) Click on the Address field in the Group Name column.
- 4) Click on the X's associated with the unused address signals to unselect them.
- 5) File->Save System As... into c:\program files\TLA 700\Supports\ARM-EBI\ARM-EBI.tla (note this will overwrite the original ARM-EBI system setup).

## **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.