



## **AMX™ MA32 Tool Guide**

**First Printing: June 1, 2001**  
**Last Printing: March 1, 2005**

**Copyright © 2001 - 2005**

**KADAK Products Ltd.**  
**206 - 1847 West Broadway Avenue**  
**Vancouver, BC, Canada, V6J 1Y5**  
**Phone: (604) 734-2796**  
**Fax: (604) 734-8114**



## TECHNICAL SUPPORT

KADAK Products Ltd. is committed to technical support for its software products. Our programs are designed to be easily incorporated in your systems and every effort has been made to eliminate errors.

Engineering Change Notices (ECNs) are provided periodically to repair faults or to improve performance. You will automatically receive these updates during the product's initial support period. For technical support beyond the initial period, you must purchase a Technical Support Subscription. Contact KADAK for details. Please keep us informed of the primary user in your company to whom update notices and other pertinent information should be directed.

Should you require direct technical assistance in your use of this KADAK software product, engineering support is available by telephone, fax or e-mail. KADAK reserves the right to charge for technical support services which it deems to be beyond the normal scope of technical support.

We would be pleased to receive your comments and suggestions concerning this product and its documentation. Your feedback helps in the continuing product evolution.

KADAK Products Ltd.  
206 - 1847 West Broadway Avenue  
Vancouver, BC, Canada, V6J 1Y5

Phone: (604) 734-2796  
Fax: (604) 734-8114  
e-mail: [amxtech@kadak.com](mailto:amxtech@kadak.com)

**Copyright © 2001-2005 by KADAK Products Ltd.  
All rights reserved.**

No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual or otherwise, without the prior written permission of KADAK Products Ltd., Vancouver, B.C., CANADA.

### **DISCLAIMER**

KADAK Products Ltd. makes no representations or warranties with respect to the contents hereof and specifically disclaims any implied warranties of merchantability and fitness for any particular purpose. Further, KADAK Products Ltd. reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation of KADAK Products Ltd. to notify any person of such revision or changes.

### **TRADEMARKS**

AMX in the stylized form and KwikNet are registered trademarks of KADAK Products Ltd. AMX, AMX/FS, InSight, *KwikLook* and KwikPeg are trademarks of KADAK Products Ltd. Microsoft, MS-DOS and Windows are registered trademarks of Microsoft Corporation. MIPS32 is a trademark of MIPS Technologies, Inc. All other trademarked names are the property of their respective owners.

**AMX MA32 TOOL GUIDE**  
**Table of Contents**

	<b>Page</b>
<b>1. Selecting a Tool Set</b>	<b>1-1</b>
<b>2. MetaWare (MW) Tool Guide</b>	<b>2-1</b>

This page left blank intentionally.

# 1. Selecting a Tool Set

## Available Toolsets

AMX™ MA32 (for MIPS32 processors) and the *KwikLook*™ Fault Finder have been developed on a PC with Microsoft® Windows® using the software development tools described in this guide.

To simplify the selection process, KADAK has prepared this Tool Guide. This chapter introduces the tools and defines the subsets which KADAK has used with success. Subsequent chapters provide specific guidelines for using each of the supported toolset combinations with AMX MA32.

Note that AMX MA32 is delivered to you ready to use with each of the supported toolsets. Should you wish to rebuild the AMX MA32 Library for any reason, follow the construction guidelines provided in Appendix D of the AMX User's Guide.

To construct your embedded application, you will require a C or C++ compiler, an assembler, a librarian (optional), a linker and/or locator and a remote debugger. The vendors listed below provide these tools. The tool name listed is the vendor's product name or the name of the executable program used to run the tool. The tool name listed will be used throughout this manual to reference the specific tool from a particular vendor.

<b>Vendor</b>	<b>C/C++</b>	<b>Assembler</b>	<b>Librarian</b>	<b>Linker</b>	<b>Locator</b>	<b>Debugger</b>
MetaWare	<i>HCMIPS</i>	<i>ASMIPS</i>	<i>ARMIPS</i>	<i>LDMIPS</i>	<i>LDMIPS</i>	<i>SeeCode</i>

## Supported Toolsets

Unfortunately you cannot arbitrarily use any combination of the listed tools. Of all the tools listed, KADAK has identified the combinations which can be used with AMX MA32. The supported toolsets are divided into major classes according to the C/C++ compiler vendor and then, if necessary, into sub-classes, one for each locator and/or debugger.

Each supported toolset is given a three character mnemonic called a **toolset id** which is used by KADAK to identify the toolset combination. The first two characters of the mnemonic identify the compiler vendor. The third character, if needed, identifies the locator and/or debugger used.

**Compiler**  
*MW* MetaWare Incorporated High C/C++

**Debugger**  
-- MetaWare SeeCode for MIPS32

The following toolset combinations are supported by KADAK.

Toolset id: **MW**  
Vendor: MetaWare

C/C++ *HCMIPS*  
Assembler *ASMIPS*  
Librarian *ARMIPS*  
Linker/ *LDMIPS*  
Locator  
Debugger *SeeCode*

## 2. MetaWare (MW) Tool Guide

AMX™ MA32 (for MIPS32 processors) has been developed on a PC with Windows® NT v4.0 using the MetaWare tools listed below. The AMX libraries and object modules on the product disks have been generated using the most recent tools listed. If you are not using this toolset, you may have to rebuild the AMX libraries in order to use your out-of-date tools.

<b>MetaWare Tools</b>		<b><u>v4.5</u></b>	<b><u>v4.5a</u></b>
<i>HCMIPS</i>	MIPS32 High C/C++ compiler	R4.5	R4.5a15
<i>ASMIPS</i>	MIPS32 assembler	V1.3	V1.3c
<i>ARMIPS</i>	MIPS32 archiver (librarian)	2.12	2.12
<i>LDMIPS</i>	MIPS32 linker/locator	5.1k	5.3b
	SeeCode MIPS32 Debugger	v4.5	v6.0

AMX MA32 and *KwikLook* have been tested on the following platforms.

MIPS:

Malta 4Kc Development Board

### Environment Variables

Set the following environment variables to provide access to all AMX and MetaWare tools, header files, object files and libraries.

<i>CJPATH</i>	Path to AMX installation directory (. . . \AMX442)
<i>PATH</i>	Path to AMX and MetaWare executable programs
<i>TOOLS DIR</i>	Path to MetaWare executable programs
<i>HCDIR</i>	Path to MetaWare installation directory
<i>IPATH</i>	Path to MetaWare C include files
<i>LIBPATH</i>	Path to MetaWare C library files
<i>TMP , TMPPREFIX</i>	Path to a temporary directory for use by MetaWare tools

The AMX libraries have been constructed using the following assumptions. The resulting AMX MA32 libraries are ready for use with all MIPS32 implementations.

AMX MA32 target processor is a little endian generic MIPS32  
AMX MA32 is generated in ELF object format  
AMX MA32 assumes software floating point emulation

## Object Formats

MetaWare only supports the ELF object format. The AMX MA32 libraries and object modules are provided in ELF format. Your object modules and the AMX and MetaWare libraries and object modules, all in ELF format, can be combined to create an executable module in ELF format suitable for use with the MetaWare SeeCode Debugger.

## Parameter Passing Conventions

The MetaWare tools only support the MIPS32 C function parameter passing convention. AMX MA32 follows the MIPS32 standard, the parameter passing convention common to all toolsets supported by KADAK.

### Warning

Any AMX message exchange task which receives floating point parameters within an AMX message **MUST** be built to receive AMX messages by reference, **NOT** by value.

## Register Usage

The MetaWare version of AMX makes the following C interface register assumptions. Registers *at*, *v0*, *v1*, *a0-a3*, *t0-t7*, *t8* and *t9* can always be altered by C procedures. Registers *s0-s7*, *s8*, *gp* and *sp* are preserved by AMX according to the MetaWare rules for C procedures. Registers *k0* and *k1* are reserved for use by the AMX Exception Supervisor. Integers and pointers are returned from C procedures in register *v0*. Register *gp* is dedicated for global data access. You must **NOT** use any C compilation switch which changes these register assumptions.

## Big or Little Endian

AMX MA32 is delivered ready for use with the little endian model. AMX MA32 will also operate, without modification, on big endian hardware. However, to use AMX on big endian hardware, you must first **rebuild the AMX Library for big endian operation**. You must also be sure to use the MetaWare big endian C startup code and libraries.

To rebuild the AMX Library for big endian operation, set environment variable *AMX\_ENDN=B*. If you must rebuild the AMX Library for little endian operation, set environment variable *AMX\_ENDN=L* or leave it undefined. Then build the library as described in Appendix D of the AMX User's Guide.

To use the little endian model, you must set *BE* to *0* in the configuration register (*CP0* register 16) prior to launching AMX. To use the big endian model, you must set *BE* to *1*. The state of *BE* is usually set by hardware when the processor is reset.

## Using the MetaWare C Compiler

All AMX header files *CJ442xxx.H* and the generic AMX include file *CJZZZ.H* must be present in the current directory together with your source file being compiled.

Use the following compilation switches when you are compiling modules for use in the AMX environment.

	by default	; no stack checking
	by default	; output object module <i>FILENAME.O</i>
	<i>-Hefile=FILENAME.ERR</i>	; redirect C error messages to <i>FILENAME.ERR</i>
	<i>-c</i>	; compile only
	<i>-HL</i>	; compile little endian ( <i>-HB</i> for big endian)
	<i>-mips32</i>	; MIPS32 instruction compliance
	<i>-Hthread</i>	; compile for thread-safe use (preferred)
	or	
	<i>-Hnothread</i>	; compile for non-thread-safe use
	<i>-O</i>	; full optimize for speed
	<i>-g</i>	; (optional) generate debug information

The compilation command line is therefore of the form:

```
HCMIPS -c -HL -mips32 -Hthread -O FILENAME.C -Hefile=FILENAME.ERR
```

The following command line switches, although optional, are recommended.

```
-Hpragma=Offwarn(572) ; disable dangerous typecast warnings
```

## Compiling the AMX System Configuration Module

Your AMX System Configuration Module *SYSCFG.C* is compiled as follows. All AMX header files *CJ442xxx.H* and the generic AMX include file *CJZZZ.H* must be present in the current directory together with file *SYSCFG.C*.

Use the *-HL* switch for little endian systems and *-HB* for big endian systems. Use the *-Hthread* switch for thread-safe systems and *-Hnothread* for non-threaded systems.

```
HCMIPS -c -HL -mips32 -Hthread -O SYSCFG.C -Hefile=SYSCFG.ERR
```

## Assembling the AMX Target Configuration Module

Your AMX Target Configuration Module *HDWCFG.S* is assembled as follows. The generic AMX header file *CJZZZK.DEF* must be present in the current directory together with file *HDWCFG.S*.

Use the following command line switches when using the MetaWare assembler.

```
by default          ; assemble with case sensitivity
by default          ; output object module HDWCFG.O
>HDWCFG.ERR        ; redirect assembler error messages to HDWCFG.ERR
-le                 ; assemble little endian (-be for big endian)
-mips32             ; MIPS32 instruction compliance
-noat               ; warn if macro uses register $at
-noreorder          ; no instruction reordering
```

```
ASMIPS -le -mips32 -noat -noreorder HDWCFG.S >HDWCFG.ERR
```

## Making Libraries

To make a library from a collection of object modules, create a library specification file *YOURLIB.LBM*. Use the MetaWare version of the AMX library specification file *CJ442.LBM* as a guide.

Use the following command line switches when using the MetaWare librarian.

```
-r                 ; replace (add) modules in (to) library
YOURLIB.A         ; output library module YOURLIB.A
>YOURLIB.LBE      ; redirect librarian error messages to YOURLIB.LBE
```

Make your library as follows.

```
ARMIPS -r YOURLIB.A @YOURLIB.LBM >YOURLIB.LBE
```

## Building the AMX Library

When rebuilding the AMX Library using MetaWare tools, you must define the following environment variables. To build the library for little endian operation, set environment variable *AMX\_ENDN=L* or leave it undefined. To build the library for big endian operation, set environment variable *AMX\_ENDN=B*.

To build the AMX Library with support for MetaWare's thread-safe library, set environment variable *AMX\_TSAFE=TSOON* or leave it undefined. To build the library without such thread-safe support, set environment variable *TSAFE=TSOFF*. The library is then built as described in Appendix D of the AMX User's Guide.

## Thread-safe Linking with the MetaWare Linker

When used with MetaWare C thread-safe libraries, the modules which form your AMX system must be linked in the following order.

Your *MAIN* module

Other application modules

<i>SYSCFG.O</i>	; AMX System Configuration Module
<i>HDWCFG.O</i>	; AMX Target Configuration Module
<i>CHxxxxxT.O</i>	; AMX chip-specific clock driver or your equivalent
<i>CJ442UF.O</i>	; Launch and leave AMX (may be customized)
<i>CJ442RAC.O</i>	; AMX ROM Access Module (customized) ; (only if AMX placed in a separate ROM) ; (see Appendix C in AMX MA32 Target Guide)
<i>CJ442CV.A</i>	; AMX MA32 vc Conversion Library ; (only if converting an AMX 86 v3, AMX 386 v1 or ; AMX 68000 v2 application)
<i>CJ442.A</i>	; AMX MA32 Library ; The compiler driver will select the required ; <i>CRT1.O</i> startup module and libraries automatically. ; However, MetaWare C Thread-safe Runtime Libraries ; <i>LIBCT.A</i> and <i>LIBMWT.A</i> can be specified on the link ; command line (see link example).

### Note

Symbol *cj\_mw\_tls* (or *cj\_mw\_tlsmin*) must be undefined (using the linker *-u* switch) to force the full (or minimal) thread-safe support modules to be resolved from the AMX MA32 Library.

Be sure to follow the guidelines for thread-safe library usage presented later in this Tool Guide.

## Non-threaded Linking with the MetaWare Linker

When used with the non-threaded MetaWare C libraries, the modules which form your AMX system must be linked in the following order.

Your *MAIN* module

Other application modules

<i>SYSCFG.O</i>	; AMX System Configuration Module
<i>HDWCFG.O</i>	; AMX Target Configuration Module
<i>CHxxxxxT.O</i>	; AMX chip-specific clock driver or your equivalent
<i>CJ442UF.O</i>	; Launch and leave AMX (may be customized)
<i>CJ442RAC.O</i>	; AMX ROM Access Module (customized) ; (only if AMX placed in a separate ROM) ; (see Appendix C in AMX MA32 Target Guide)
<i>CJ442CV.A</i>	; AMX MA32 vc Conversion Library ; (only if converting an AMX 86 v3, AMX 386 v1 or ; AMX 68000 v2 application)
<i>CJ442.A</i>	; AMX MA32 Library ; The compiler driver will select the required ; <i>CRT1.O</i> startup module and libraries automatically. ; However, MetaWare C Non-threaded Runtime Libraries ; <i>LIBC.A</i> and <i>LIBMW.A</i> can be specified on the link ; command line (see link example).

Create a link specification file *YOURLINK.LKS*. Use the MetaWare version of the AMX Sample Program link specification file *CJSAMPLE.LKS* as a guide.

Start with the sample link specification file for the board which most closely resembles your hardware configuration.

Note

If you decide to omit any of the link and locate commands from the sample specification, you may encounter link errors or run-time faults.

Link and locate with the MetaWare linker using the following command line switches.

```
-Hthread          ; link for thread-safe use (preferred)
-u cj_mw_tls      ; load AMX MA32 thread-safe support (full)
-u cj_mw_tlsmin   ; load AMX MA32 thread-safe support (minimal)
                  ; (see the topic Guidelines for Thread-safe Library Use)
or
-Hnothread        ; link for non-thread-safe use

-m               ; generate section and symbol map
-HL              ; link little endian (-HB for big endian)
-o YOURLINK.OUT  ; direct link output to file YOURLINK.OUT
YOURLINK.LKS     ; use link specification file YOURLINK.LKS
-fsoft           ; use floating point emulation libraries
-Lmwpath         ; path to mwpath library directory (for -lxxxx searches)
-lxxxx          ; link library named LIBXXXX.A
>YOURLINK.MAP    ; direct section and symbol map to file YOURLINK.MAP

-o YOURLINK.HEX ; create hex output in file YOURLINK.HEX
                  ; generate Motorola S-record format
                  ; other formats can be generated
                  ; (see MetaWare manual)
```

The compiler driver's link and locate command line is therefore of the following form.

For little endian use with full thread-safe support, link as follows:

```
HCMIPS -o YOURLINK.OUT -m -fsoft -HL -Hthread -u cj_mw_tls
        YOURLINK.LKS -Lmwpath -lct -lmwt >YOURLINK.MAP
ELF2HEX -o YOURLINK.HEX YOURLINK.OUT
```

The resulting load module *YOURLINK.OUT* is suitable for use with the MetaWare SeeCode MIPS32 debugger. The load module *YOURLINK.HEX* is ready for burning into EPROM.

For little endian use with no thread-safe support, link as follows:

```
HCMIPS -o YOURLINK.OUT -m -fsoft -HL -Hnothread
        YOURLINK.LKS -Lmwpath -lc -lmw >YOURLINK.MAP
ELF2HEX -o YOURLINK.HEX YOURLINK.OUT
```

The resulting load module *YOURLINK.OUT* is suitable for use with the MetaWare SeeCode MIPS32 debugger. The load module *YOURLINK.HEX* is ready for burning into EPROM.

## Guidelines for Thread-safe Library Use

KADAK provides support for the thread-safe run-time libraries included with the MetaWare High C/C++ MIPS32 Embedded Development tools. You only need to use the thread-safe libraries if your application has multiple tasks which concurrently use the library functions which can benefit from thread-safe execution. For example, if only one task does floating point operations, there may be no need to use the thread-safe libraries.

Support for the thread-safe MetaWare libraries is built into the AMX MA32 Library. The following files in the AMX MA32 Library satisfy the kernel-dependent requirements of the MetaWare thread-safe libraries.

<i>AMX_LIST.C</i>	Low-level list manipulation
<i>AMX_MUTX.C</i>	Kernel-dependent mutex implementation
<i>AMX_TLS.C</i>	Thread-local storage allocation and freeing
<i>AMX_TLTK.C</i>	Thread-safe <i>errno</i> support
<i>AMX_TLSX.C</i>	Mutex stubs for minimal thread-safe support

Add one of the following statements to your link specification file or to your link command line to force the appropriate thread-safe support modules to be loaded from the AMX MA32 Library.

```
-u cj_mw_tls           ; load AMX MA32 thread-safe support (full)
-u cj_mw_tlsmin       ; load AMX MA32 thread-safe support (minimal)
```

MetaWare recommends that you compile and link your application modules with the *-Heos=amx* command line switch to specify AMX as the target operating system. However, you can safely omit this switch since all MetaWare specific support for AMX is provided by KADAK, not MetaWare.

The following step is very **IMPORTANT!** If you miss it, *malloc()* will fail because the *bss* and *sbss* sections are not initialized. If your link/locate specification includes a data initialization directive such as

```
INITDATA !data
```

replace it with the following line:

```
INITDATA !data, .bss, .sbss
```

## Thread-Local Storage Allocation

Thread-local storage is allocated, assigned and initialized automatically as the MetaWare Library demands. After the AMX kernel has been launched, each task that makes a call to a C library function that requires thread-local storage will be assigned its own private storage block.

If such a task is deleted with a call to AMX procedure `cjtkdelete()`, the task's thread-local storage will not be released!

To delete such a task, you must call the new procedure `cj_mw_tkdelete()`. You will have to add the procedure's prototype in the module in which it is used:

```
CJ_ERRST CJ_CCPP cj_mw_tkdelete(CJ_ID tid, int priority)
```

Note that only one thread storage block is used by the MetaWare Library prior to the AMX launch. After AMX shuts down, the MetaWare Library will be forced to resume using the same thread storage block which was in use before AMX was launched.

## Mutex Allocation and Use

Some functions in the C/C++ run-time library must be guarded from concurrent access by multiple threads. Operations of this type include `malloc()`, `free()`, complex floating-point operations such as `log()` and, of course, the I/O library.

The thread-safe MetaWare Library uses a mutex construct to guard access to these functions. The mutex services are provided by AMX in module `AMX_MUTEX.C`.

Before AMX is launched and after AMX has shut down, mutex protection is not required. However, once AMX has been launched, AMX must ensure that the mutex protection is provided.

When a task calls a function such as `malloc()`, the AMX Semaphore Manager is called upon to create a resource semaphore to be used by the library to guard access to its memory allocation services.

Similarly, when a task calls a function such as `log()`, the AMX Semaphore Manager is called upon to create a resource semaphore to be used by the library to guard access to its floating-point services.

The AMX mutex services in file `AMX_MUTEX.C` allow a maximum of 128 concurrent mutex locks. These locks satisfy the startup and exit requirements of MetaWare. Only a small fraction of these mutex locks will be used by your AMX application and hence require the allocation of an AMX resource semaphore.

The number of resource semaphores required by the MetaWare Library depends on which library services your tasks actually use. In general, the number does not depend on how many tasks use those services.

You must adjust your AMX system configuration to include the AMX Semaphore Manager and to account for the additional semaphores that will be required.

## Fatal Thread-safe Conditions

Any of the following conditions are considered fatal:

1. A thread-storage block cannot be allocated.
2. A mutex lock is not available because the supply is exhausted.
3. An AMX semaphore cannot be created for mutex locking purposes.
4. An attempt to reference an allocated AMX semaphore was rejected by AMX.

If any of these conditions are encountered, the application will hang forever in procedure `cj_mw_fatal()` in module `AMX_MUTX.C`. When testing your application, always run with a breakpoint on `cj_mw_fatal()`. Be sure to set the breakpoint BEFORE entering the MetaWare C/C++ start-up code.

Error 1 implies that `malloc()` cannot provide the required memory.

Error 2 indicates that more than 128 mutex locks are needed.  
Adjust the definition in file `AMX_MUTX.C`.

Error 3 indicates that you need more AMX semaphores or that you may not have included the AMX Semaphore Manager in your configuration.

Error 4 usually implies that data corruption has occurred. The private AMX or MetaWare Library data structures used to manage thread-safe operation have been damaged.

## Linking a Separate AMX ROM

AMX can be committed to a separate ROM as described in Appendix C of the AMX Target Guide. Use the AMX Configuration Manager to edit your Target Parameter File *HDWCFG.UP* to define your ROM option parameters. Then use the Manager to generate your ROM Option Module *CJ442ROP.S*, ROM Access Module *CJ442RAC.S* and ROM Option link specification file *CJ442ROP.LKS*.

The ROM Option and ROM Access source modules are assembled as follows. Use the *-le* switch for little endian systems and *-be* for big endian systems.

```
ASMIPS -le -mips32 -noat -noreorder CJ442ROP.S >CJ442ROP.ERR
```

```
ASMIPS -le -mips32 -noat -noreorder CJ442RAC.S >CJ442RAC.ERR
```

The AMX ROM is linked using link specification file *CJ442ROP.LKS* as follows. Use the *-HL* switch for little endian systems and *-HB* for big endian systems. Use the *-Hthread* switch for thread-safe systems and *-Hnothread* for non-thread-safe systems. You must use the *-Hnocrt* switch to preclude linking the MetaWare C startup code and libraries.

```
HCMIPS -o AMXROM.OUT -m -HL -Hthread -Hnocrt CJ442ROP.LKS >AMXROM.MAP  
ELF2HEX -o AMXROM.HEX AMXROM.OUT
```

This example generates file *AMXROM.HEX* in Motorola S-record format suitable for transfer to ROM. Other formats supported by MetaWare can be selected with the appropriate command switch.

When you link your AMX application, be sure to include your customized AMX ROM Access Module *CJ442RAC.O* (created above) in your system link specification file.

## Using the AMX Configuration Generator

If you cannot use the AMX Configuration Manager, you may still be able to use the stand-alone AMX Configuration Generator to generate the ROM Option Module *CJ442ROP.S*, ROM Access Module *CJ442RAC.S* and ROM Option link specification file *CJ442ROP.LKS*.

Copy the ROM Option and ROM Access template files *CJ442ROP.CT* and *CJ442RAC.CT* to the current directory. Also copy the ROM Option Link Specification Template file *CJ442ROP.LKT* to the current directory.

Use the AMX Configuration Generator to generate the ROM option source modules as follows.

```
CJ442CG HDWCFG.UP CJ442ROP.CT CJ442ROP.S  
CJ442CG HDWCFG.UP CJ442RAC.CT CJ442RAC.S  
CJ442CG HDWCFG.UP CJ442ROP.LKT CJ442ROP.LKS
```

Once the ROM option source modules have been created, you can proceed to build your AMX ROM image and your AMX application as described above.

## MetaWare SeeCode Debugger

The MetaWare SeeCode™ MIPS32 Debugger supports source level debugging of your AMX MA32 system.

The SeeCode Debugger can operate by simulating a MIPS32 processor or by using a BDM connection to the MIPS32 processor.

The most effective way to use the SeeCode Debugger is to connect it to the BDM port of the target system under test using any of the wiggler devices supported by SeeCode.

## Using the *KwikLook* Fault Finder

The *KwikLook*™ Fault Finder is compatible with the SeeCode Debugger providing full screen, source level, task-aware debugging from within the Microsoft Windows® environment. *KwikLook* can be invoked directly from the debugger while at breakpoints giving you finger tip access to your application from the AMX perspective. Note that *KwikLook* and SeeCode share a common link to the target system.