



AMX[™] PPC32 Tool Guide

First Printing: June 1, 1996
Last Printing: March 1, 2005

Copyright © 1996 - 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

**Copyright © 1996-2005 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. PowerPC is a trademark of IBM Corp. All other trademarked names are the property of their respective owners.

AMX PPC32 TOOL GUIDE
Table of Contents

	Page
1. Selecting a Tool Set	1-1
2. Diab-SDS (DA) Tool Guide	2-1
3. MetaWare/SDS (MW) Tool Guide	3-1
4. Metrowerks (ME) Tool Guide	4-1

This page left blank intentionally.

1. Selecting a Tool Set

Available Toolsets

AMX™ PPC32 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 PPC32.

Note that AMX PPC32 is delivered to you ready to use with each of the supported toolsets. Should you wish to rebuild the AMX PPC32 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.

Vendor	C/C++	Assembler	Librarian	Linker	Locator	Debugger
Diab-SDS	<i>DCC</i>	<i>DAS</i>	<i>DAR</i>	<i>DLD</i>	<i>DLD</i>	<i>SingleStep</i>
MetaWare	<i>HCPPC</i>	<i>ASPPC</i>	<i>ARPPC</i>	<i>LDPPC</i>	<i>LDPPC</i>	<i>SeeCode</i>
Metrowerks	<i>MWCCEPPC</i>	<i>MWASMEPPC</i>	<i>MWLDEPPC</i>	<i>MWLDEPPC</i>	<i>MWLDEPPC</i>	<i>CodeWarrior</i>

Supported Toolsets

Unfortunately you cannot arbitrarily use any combination of the listed tools. Of all the tools listed, KADAK has identified several combinations which can be used with AMX PPC32. 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

<i>DA</i>	Diab-SDS, Inc. C/C++
<i>MW</i>	MetaWare Incorporated High C/C++
<i>ME</i>	Metrowerks Inc. C/C++

Debugger

--	Diab-SDS SingleStep for PowerPC
--	MetaWare SeeCode for PowerPC
--	Metrowerks CodeWarrior for PowerPC

The following toolset combinations are supported by KADAK.

Toolset id:	DA	MW	ME
Vendor:	Diab-SDS	MetaWare	Metrowerks
C/C++	<i>DCC</i>	<i>HCPPC</i>	<i>MWCCEPPC</i>
Assembler	<i>DAS</i>	<i>ASPPC</i>	<i>MWASMEPPC</i>
Librarian	<i>DAR</i>	<i>ARPPC</i>	<i>MWLDEPPC</i>
Linker/	<i>DLD</i>	<i>LDPPC</i>	<i>MWLDEPPC</i>
Locator			<i>MWLDEPPC</i>
Debugger	<i>SingleStep</i>	<i>SeeCode</i>	<i>CodeWarrior</i>

2. Diab-SDS (DA) Tool Guide

AMX™ PPC32 has been developed on a PC with Windows® NT v4.0 using the Diab-SDS 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.

Diab-SDS Tools		v4.1	v4.3	v4.4	v5.0	v5.1	v5.2
<i>DCC</i>	PowerPC C/C++ compiler	4.1a:0	4.3b	4.4a	5.0a	5.1.1	5.2.1
<i>DAS</i>	PowerPC assembler	4.1a:0	4.3b	4.4a	5.0a	5.1.1	5.2.1
<i>DAR</i>	PowerPC librarian	4.1a:0	4.3b	4.4a	5.0a	5.1.1	5.2.1
<i>DLD</i>	PowerPC linker	4.1a:0	4.3b	4.4a	5.0a	5.1.1	5.2.1
<i>DDUMP</i>	PowerPC locator	4.1a:0	4.3b	4.4a	5.0a	5.1.1	5.2.1
	SingleStep PowerPC Debugger	7.2	7.5	7.6	7.6	7.6	7.6
	SingleStep Target Monitor						

AMX PPC32 and *KwikLook* have been tested on the following platforms.

Motorola:

Ultra 603 Motherboard Platform
 MBX860 board
 MPC860ADS board
 MPC860FADS board
 MPC8560ADS board
 Lite5200 Evaluation Board

Others:

EST Corp SBC8260 Single Board Computer
 Embedded Planet RPX Lite MPC823 board
 IBM 403 EVB Evaluation Board
 IBM PPC405GP Reference Board
 IBM PPC440GP Reference Board

Environment Variables

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

<i>CJSPATH</i>	Path to AMX installation directory (. . . \AMX382)
<i>PATH</i>	Path to AMX and Diab-SDS executable programs
<i>TMPDIR</i>	Path to a temporary directory for use by Diab-SDS tools
<i>DIABLIB</i>	Path to Diab-SDS installation directory
<i>DTARGET</i>	Target PowerPC processor
<i>DOBJECT=E</i>	Generate ELF object format
<i>DFP=soft</i>	Assume software floating point emulation
or <i>DFP=hard</i>	Assume hardware floating point
<i>DENVIRON=cross</i>	Compiler cross-compiler on PC for PowerPC

The AMX libraries have been constructed using the following Diab-SDS parameters. Although built for the MPC603 with software floating point emulation, the resulting AMX PPC32 libraries are ready for use with all PowerPC implementations.

<i>DOBJECT=E</i>	AMX PPC32 is generated in ELF object format
<i>DTARGET=PPC603</i>	AMX PPC32 target processor is MPC603
<i>DFP=soft</i>	Assume software floating point emulation

Object Formats

Diab-SDS supports two object formats: ELF and COFF. The AMX PPC32 libraries and object modules are only provided in ELF format. Your object modules and the AMX and Diab-SDS libraries and object modules, all in ELF format, can be combined to create an executable module in ELF format suitable for use with the Diab-SDS SingleStep Debugger.

Parameter Passing Conventions

The Diab-SDS tools also support two C function parameter passing conventions: PowerPC EABI and the PowerOpen Standard. AMX PPC32 follows the EABI 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 Diab-SDS version of AMX makes the following C interface register assumptions. Registers *r0*, *r3-r12*, *lr*, *ctr* and *xer* can always be altered by C procedures. Registers *r1* and *r14-r31* are preserved by AMX according to the Diab-SDS rules for C procedures. Integers and pointers are returned from C procedures in register *r3*. Registers *r2* and *r13* are dedicated for global data access. You must NOT use any C compilation switch which changes these register assumptions.

Big or Little Endian

AMX PPC32 is delivered ready for use with the big endian model. The Diab-SDS tools do not support little endian operation. Hence, when used with Diab-SDS tools, AMX PPC32 only supports big endian operation. To use AMX PPC32 on little endian hardware, you must use one of the other supported toolsets.

If you decide to rebuild the AMX Library, select big endian operation by setting environment variable *AMX_ENDN=B* or by leaving it undefined. Build the library as described in Appendix D of the AMX User's Guide.

To use the big endian model, you must set *ILE* and *LE* to 0 in the machine state register (*MSR*) prior to launching AMX. This is the default state when the processor is reset.

Using the Diab-SDS C Compiler

All AMX header files *CJ382xxx.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>-@E=</i>	; redirect C error messages to <i>FILENAME.ERR</i>
	<i>-c</i>	; compile only
	<i>-XO</i>	; full optimize for speed
	<i>-g</i>	; (optional) generate debug information

The compilation command line is therefore of the form:

```
DCC -c -XO FILENAME.C -@E=FILENAME.ERR
```

Compiling the AMX System Configuration Module

Your AMX System Configuration Module *SYSCFG.C* is compiled as follows. All AMX header files *CJ382xxx.H* and the generic AMX include file *CJZZZ.H* must be present in the current directory together with file *SYSCFG.C*.

```
DCC -c -XO SYSCFG.C -@E=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*.

The Diab-SDS C command line driver is used to invoke the assembler. Some of the command line switches match those used for C. Others are as follows.

by default	; assemble with case sensitivity
by default	; output object module <i>HDWCFG.O</i>
-@E=	; redirect assembler error messages to <i>HDWCFG.ERR</i>
-Wa, -x	; omit local symbols from object module
-c	; assemble only

```
DCC -c -Wa, -x HDWCFG.S -@E=HDWCFG.ERR
```

Making Libraries

To make a library from a collection of object modules, create a library specification file *YOURLIB.LBM*. Use the Diab-SDS version of the AMX library specification file *CJ382.LBM* as a guide.

Use the following command line switches when using the Diab-SDS librarian.

<i>YOURLIB.A</i>	; output library module <i>YOURLIB.A</i>
> <i>YOURLIB.LBE</i>	; redirect librarian error messages to <i>YOURLIB.LBE</i>
-qc	; create a new library; use quick append mode

Make your library as follows.

```
DAR -qc YOURLIB.A -@YOURLIB.LBM >YOURLIB.LBE
```

Linking with the Diab-SDS Linker

When used with Diab-SDS C, 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 ; (not required if PowerPC decremter used as clock)
<i>CJ382UF.O</i>	; Launch and leave AMX (may be customized)
<i>CJ382RAC.O</i>	; AMX ROM Access Module (customized) ; (only if AMX placed in a separate ROM) ; (see Appendix C in AMX PPC32 Target Guide)
<i>CJ382CV.A</i>	; AMX PPC32 vc Conversion Library ; (only if converting an AMX 86 v3, AMX 386 v1 or ; AMX 68000 v2 application)
<i>CJ382.A</i>	; AMX PPC32 Library

Diab-SDS C Runtime Libraries for target hardware

Create a link specification file *YOURLINK.LKS*. Use the Diab-SDS 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 Diab-SDS linker and locator using the following command line switches.

```
-m                ; create section summary
-Wm              ; no default link command file
-o              ; direct link output to file YOURLINK.OUT
-@E=            ; direct link error messages to file YOURLINK.LKE
>YOURLINK.MAP   ; direct section summary to file YOURLINK.MAP

-t              ; create summary of symbol values
-v              ; inhibit output of .bss section to minimize
                ; the size of the resulting HEX file.
-R              ; generate Motorola S-record format
                ; other formats can be generated
                ; (see Diab-SDS manual)
-o              ; direct locate output to file YOURLINK.HEX
>YOURLINK.SYM  ; direct symbol summary to file YOURLINK.SYM
```

The link and locate command lines are therefore of the form:

```
DCC -m -Wm -o YOURLINK.OUT YOURLINK.LKS -@E=YOURLINK.LKE >YOURLINK.MAP
DDUMP -t -v -R -o YOURLINK.HEX YOURLINK.OUT >YOURLINK.SYM
```

The resulting load module *YOURLINK.OUT* is suitable for use with the Diab-SDS SingleStep PowerPC debugger.

The resulting load module *YOURLINK.HEX* is ready for burning into EPROM.

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 *CJ382ROP.S*, ROM Access Module *CJ382RAC.S* and ROM Option link specification file *CJ382ROP.LKS*.

The ROM Option and ROM Access source modules are assembled as follows.

```
DCC -c -Wa,-x CJ382ROP.S -@E=CJ382ROP.ERR
```

```
DCC -c -Wa,-x CJ382RAC.S -@E=CJ382RAC.ERR
```

The AMX ROM is linked using link specification file *CJ382ROP.LKS* as follows.

```
DCC -m -Wm -Ws -Wc -o AMXROM.OUT CJ382ROP.LKS  
-@E=AMXROM.LKE >AMXROM.MAP  
DDUMP -t -v -R -o AMXROM.HEX AMXROM.OUT >AMXROM.SYM
```

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

Note that command line switch *-Ws* is used to prevent loading of the default C startup module. Command line switch *-Wc* is used to prevent loading of the default C runtime library.

When you link your AMX application, be sure to include your customized AMX ROM Access Module *CJ382RAC.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 *CJ382ROP.S*, ROM Access Module *CJ382RAC.S* and ROM Option link specification file *CJ382ROP.LKS*.

Copy the ROM Option and ROM Access template files *CJ382ROP.CT* and *CJ382RAC.CT* to the current directory. Also copy the ROM Option Link Specification Template file *CJ382ROP.LKT* to the current directory.

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

```
CJ382CG HDWCFG.UP CJ382ROP.CT CJ382ROP.S  
CJ382CG HDWCFG.UP CJ382RAC.CT CJ382RAC.S  
CJ382CG HDWCFG.UP CJ382ROP.LKT CJ382ROP.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.

Diab-SDS SingleStep Debugger

The Diab-SDS SingleStep™ PowerPC Debugger supports source level debugging of your AMX PPC32 system.

The SingleStep Debugger can operate by simulating a PowerPC or by using a BDM connection to the PowerPC.

SingleStep can also operate using a serial (or other) connection to the target PowerPC system under test. When used in this fashion, you must install the SingleStep Target Monitor in your target hardware. Instructions for doing so are provided in the SingleStep Reference Manual. Your version of the SingleStep Target Monitor must provide a device driver for the serial (or other) device used for communication with the SingleStep Debugger. It is recommended that your driver use polled I/O so that the SingleStep Target Monitor can operate with interrupts disabled.

Using the *KwikLook* Fault Finder

The *KwikLook*™ Fault Finder is compatible with the SingleStep 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 SingleStep share a common link to the target system.

3. MetaWare (MW) Tool Guide

AMX™ PPC32 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.

MetaWare Tools	<u>v4.1</u>	<u>v4.3c</u>	<u>v4.5</u>	<u>v4.5c</u>	
<i>HCPPC</i>	PowerPC High C/C++ compiler	4.1a	R4.3C	R4.5a	R4.5c
<i>ASPPC</i>	PowerPC assembler	2.39	V3.3	V3.13a	V3.25
<i>ARPPC</i>	PowerPC archiver (librarian)	2.07	2.10	2.10	2.12
<i>LDPPC</i>	PowerPC linker/locator	4.8t	5.0v	5.1f	5.3b
	SeeCode PowerPC Debugger		v4.3	v4.5	v6.0

AMX PPC32 and *KwikLook* have been tested on the following platforms.

Motorola:

Ultra 603 Motherboard Platform
MBX860 board
MPC860ADS board
MPC860FADS board
MPC8560ADS board
Lite5200 Evaluation Board

Others:

EST Corp SBC8260 Single Board Computer
Embedded Planet RPX Lite MPC823 board
IBM 403 EVB Evaluation Board
IBM PPC405GP Reference Board
IBM PPC440GP Reference 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 (. . . \AMX382)
<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 PPC32 libraries are ready for use with all PowerPC implementations.

AMX PPC32 target processor is a big endian generic PowerPC
AMX PPC32 is generated in ELF object format
AMX PPC32 assumes software floating point emulation

Object Formats

MetaWare only supports the ELF object format. The AMX PPC32 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 PowerPC EABI C function parameter passing convention. AMX PPC32 follows the EABI 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 *r0*, *r3-r12*, *lr*, *ctr* and *xer* can always be altered by C procedures. Registers *r1* and *r14-r31* are preserved by AMX according to the MetaWare rules for C procedures. Integers and pointers are returned from C procedures in register *r3*. Registers *r2* and *r13* are dedicated for global data access. You must **NOT** use any C compilation switch which changes these register assumptions.

Big or Little Endian

AMX PPC32 is delivered ready for use with the big endian model. AMX PPC32 will also operate, without modification, on little endian hardware. However, to use AMX on little endian hardware, you must first **rebuild the AMX Library for little endian operation**. You must also be sure to use the MetaWare little endian C startup code and libraries.

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

To use the big endian model, you must set *ILE* and *LE* to 0 in the machine state register (*MSR*) prior to launching AMX. This is the default state when the processor is reset. To use the little endian model, you must set both *ILE* and *LE* to 1.

Using the MetaWare C Compiler

All AMX header files *CJ382xxx.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.

<i>by default</i>	; no stack checking
<i>by default</i>	; output object module <i>FILENAME.O</i>
<i>>FILENAME.ERR</i>	; redirect C error messages to <i>FILENAME.ERR</i>
<i>-c</i>	; compile only
<i>-HB</i>	; compile big endian (<i>-HL</i> for little endian)
<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:

```
HCPPC -c -HB -Hthread -O FILENAME.C >FILENAME.ERR
```

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

<i>-Hon=Use_eieio</i>	; insert <i>EIEIO</i> instructions before volatile references
<i>-Hpragma=Offwarn(572)</i>	; disable dangerous typecast warnings

The following command line switch must be used for processors such as the MPC8560 which are based on the Freescale PowerPC e500 core. The command prevents the compiler from generating the *LSWx* and *STSWx* instructions which the e500 core does not support.

```
-Hoff=Use_loadstore_string
```

Compiling the AMX System Configuration Module

Your AMX System Configuration Module *SYSCFG.C* is compiled as follows. All AMX header files *CJ382xxx.H* and the generic AMX include file *CJZZZ.H* must be present in the current directory together with file *SYSCFG.C*.

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

```
HCPPC -c -HB -Hthread -O SYSCFG.C >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
-be                 ; assemble big endian (-le for little endian)
-big_si             ; suppress big integer warnings

ARPPC -be -big_si 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 *CJ382.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.

```
ARPPC -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 big endian operation, set environment variable *AMX_ENDN=B* or leave it undefined. To build the library for little endian operation, set environment variable *AMX_ENDN=L*.

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 ; (not required if PowerPC decremter used as clock)
<i>CJ382UF.O</i>	; Launch and leave AMX (may be customized)
<i>CJ382RAC.O</i>	; AMX ROM Access Module (customized) ; (only if AMX placed in a separate ROM) ; (see Appendix C in AMX PPC32 Target Guide)
<i>CJ382CV.A</i>	; AMX PPC32 vc Conversion Library ; (only if converting an AMX 86 v3, AMX 386 v1 or ; AMX 68000 v2 application)
<i>CJ382.A</i>	; AMX PPC32 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 PPC32 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 ; (not required if PowerPC decremter used as clock)
<i>CJ382UF.O</i>	; Launch and leave AMX (may be customized)
<i>CJ382RAC.O</i>	; AMX ROM Access Module (customized) ; (only if AMX placed in a separate ROM) ; (see Appendix C in AMX PPC32 Target Guide)
<i>CJ382CV.A</i>	; AMX PPC32 vc Conversion Library ; (only if converting an AMX 86 v3, AMX 386 v1 or ; AMX 68000 v2 application)
<i>CJ382.A</i>	; AMX PPC32 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).

Note

The AMX C startup module *CJ382SU.O* previously used to replace the MetaWare *CRT1.O* module is no longer used.

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 PPC32 thread-safe support (full)
-u cj_mw_tlsmin    ; load AMX PPC32 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
-HB               ; link big endian (-HL for little 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
-Lmwwpath         ; path to mwwpath 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 big endian use with full thread-safe support, link as follows:

```
HCPPC -o YOURLINK.OUT -m -fsoft -HB -Hthread -u cj_mw_tls
      YOURLINK.LKS -Lmwwpath -lct -lmwt >YOURLINK.MAP
ELF2HEX -o YOURLINK.HEX YOURLINK.OUT
```

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

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

```
HCPPC -o YOURLINK.OUT -m -fsoft -HB -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 PowerPC 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++ PowerPC 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.

The text files provided with MetaWare High C/C++ release 4.3c and earlier describe how the earliest releases of AMX PPC32 could be used with the thread-safe libraries. Starting with AMX PPC32 v1.04a, support for the thread-safe libraries is built into the AMX PPC32 Library. The following files in the AMX PPC32 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_TLSK.C</i>	Thread-safe <i>errno</i> support
<i>AMX_TLSX.C</i>	Mutex stubs for minimal thread-safe support

Add one 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 PPC32 Library.

```
-u cj_mw_tls      ; load AMX PPC32 thread-safe support (full)
-u cj_mw_tlsmin   ; load AMX PPC32 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 *CJ382ROP.S*, ROM Access Module *CJ382RAC.S* and ROM Option link specification file *CJ382ROP.LKS*.

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

```
ASPPC -be -big_si CJ382ROP.S >CJ382ROP.ERR
```

```
ASPPC -be -big_si CJ382RAC.S >CJ382RAC.ERR
```

The AMX ROM is linked using link specification file *CJ382ROP.LKS* as follows. Use the *-HB* switch for big endian systems and *-HL* for little 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.

```
HCPPC -o AMXROM.OUT -m -HB -Hthread -Hnocrt CJ382ROP.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 *CJ382RAC.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 *CJ382ROP.S*, ROM Access Module *CJ382RAC.S* and ROM Option link specification file *CJ382ROP.LKS*.

Copy the ROM Option and ROM Access template files *CJ382ROP.CT* and *CJ382RAC.CT* to the current directory. Also copy the ROM Option Link Specification Template file *CJ382ROP.LKT* to the current directory.

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

```
CJ382CG HDWCFG.UP CJ382ROP.CT CJ382ROP.S  
CJ382CG HDWCFG.UP CJ382RAC.CT CJ382RAC.S  
CJ382CG HDWCFG.UP CJ382ROP.LKT CJ382ROP.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™ PowerPC Debugger supports source level debugging of your AMX PPC32 system.

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

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.

4. Metrowerks (ME) Tool Guide

AMX™ PPC32 has been developed on a PC with Windows® NT v4.0 using the Metrowerks 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.

Metrowerks Tools		<u>v5.0</u>	<u>v6.0</u>	v6.1	v8.0
				<u>v6.5</u>	<u>v6.6</u>
<i>MWCCEPPC</i>	PowerPC C/C++ compiler	2.3.3	2.4.2	2.4.2	3.0.4
<i>MWASMEPPC</i>	PowerPC assembler	2.3.2	2.4.2	2.4.2	3.0.4
<i>MWLDEPPC</i>	PowerPC linker/librarian	2.3.3	2.4.8	2.4.8	3.0.4
	CodeWarrior IDE				
	CodeWarrior PowerPC Debugger				
	<i>MetroTRK</i> Target Resident Kernel				

AMX PPC32 and *KwikLook* have been tested on the following platforms.

Motorola:

Ultra 603 Motherboard Platform
 MBX860 board
 MPC860ADS board
 MPC860FADS board
 MPC8560ADS board
 Lite5200 Evaluation Board

Others:

EST Corp SBC8260 Single Board Computer
 Embedded Planet RPX Lite MPC823 board
 IBM 403 EVB Evaluation Board
 IBM PPC405GP Reference Board
 IBM PPC440GP Reference Board

Environment Variables

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

<i>CJPATH</i>	Path to AMX installation directory (... \AMX382)
<i>PATH</i>	Path to AMX and Metrowerks executable programs
<i>TMPDIR</i>	Path to a temporary directory for use by Metrowerks tools
<i>CWFolder</i>	Path to Metrowerks installation directory
<i>MWCIncludes</i>	Path to Metrowerks include directories
<i>MWLibraries</i>	Path to Metrowerks library directories
<i>MWLibraryFiles</i>	Filenames of Metrowerks C libraries to be searched

The AMX libraries have been constructed using the following assumptions. The resulting AMX PPC32 libraries are ready for use with all PowerPC implementations.

AMX PPC32 target processor is a big endian generic PowerPC
 AMX PPC32 is generated in ELF object format
 AMX PPC32 assumes software floating point emulation

Command Line Tools

The Metrowerks CodeWarrior Integrated Development Environment (IDE) provides a software development environment within which you can readily create a project which incorporates AMX. However, the AMX library construction process is independent of the CodeWarrior IDE.

To make the AMX libraries and to construct an AMX application as described in this Tool Guide, you must use the Metrowerks command line tools. It is assumed that the following Metrowerks tools have been copied from the Metrowerks installation directory to the Metrowerks *BIN* directory and renamed as follows.

<code>... \PowerPC_EABI_Tools \Command_Line_Tools \mwcceppc.exe</code>	to <code>MWCC.EXE</code>
<code>... \PowerPC_EABI_Tools \Command_Line_Tools \mwasmepc.exe</code>	to <code>MWASM.EXE</code>
<code>... \PowerPC_EABI_Tools \Command_Line_Tools \mwldeppc.exe</code>	to <code>MWLD.EXE</code>

Object Formats

Metrowerks supports the ELF object format. The AMX PPC32 libraries and object modules are provided in ELF format. Your object modules and the AMX and Metrowerks libraries and object modules, all in ELF format, can be combined to create an executable module in ELF format suitable for use with the Metrowerks CodeWarrior Debugger.

Parameter Passing Conventions

The Metrowerks tools support the PowerPC EABI C function parameter passing conventions. AMX PPC32 follows the EABI 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 Metrowerks version of AMX makes the following C interface register assumptions. Registers *r0*, *r3-r12*, *lr*, *ctr* and *xer* can always be altered by C procedures. Registers *r1* and *r14-r31* are preserved by AMX according to the Metrowerks rules for C procedures. Integers and pointers are returned from C procedures in register *r3*. Registers *r2* and *r13* are dedicated for global data access. You must **NOT** use any C compilation switch which changes these register assumptions.

Big or Little Endian

AMX PPC32 is delivered ready for use with the big endian model. AMX PPC32 will also operate, without modification, on little endian hardware. However, to use AMX on little endian hardware, you must first **rebuild the AMX Library for little endian operation**. Since Metrowerks does not provide prebuilt little endian libraries, you must first create a little endian version of the Metrowerks C startup code and libraries for use with AMX.

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

To use the big endian model, you must set `ILE` and `LE` to 0 in the machine state register (`MSR`) prior to launching AMX. This is the default state when the processor is reset. To use the little endian model, you must set both `ILE` and `LE` to 1.

Using the Metrowerks C Compiler

All AMX header files `CJ382xxx.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
	<code>-o</code>	; output object module <code>FILENAME.O</code>
	<code>>FILENAME.ERR</code>	; redirect C error messages to <code>FILENAME.ERR</code>
	<code>-c</code>	; compile only
	<code>-big</code>	; compile big endian (<code>-little</code> for little endian)
	<code>-Op</code>	; optimize for speed
	<code>-g</code>	; (optional) generate debug information

The following compilation switches are used when building AMX and may be suitable for compiling your application modules as well.

	<code>-rostr</code>	; make string constants read only
	<code>-sdata 0</code>	; small mutable data section is empty
	<code>-sdata2 0</code>	; small constant data section is empty
	<code>-Cpp_exceptions off</code>	; no C++ exception handling needed

If compiling for little endian operation, the following compilation switches must also be used.

	<code>-use_lmw_stmw off</code>	; no multiple word load/store for structure copying
	<code>-opt_nofunctions</code>	; no function epilogue/prologue optimizations (v5.0 only)

The compilation command line is therefore of the form:

```
MWCC -c -big -Op -o FILENAME.O FILENAME.C >FILENAME.ERR
```

Compiling the AMX System Configuration Module

Your AMX System Configuration Module *SYSCFG.C* is compiled as follows. All AMX header files *CJ382xxx.H* and the generic AMX include file *CJZZZ.H* must be present in the current directory together with file *SYSCFG.C*. Use the *-big* switch for big endian systems and *-little* for little endian systems.

```
MWCC -c -big -Op -o SYSCFG.O SYSCFG.C >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*.

The Metrowerks C command line driver is used to invoke the assembler. Some of the command line switches match those used for C. Others are as follows.

by default	; assemble with case sensitivity
by default	; assemble only
<i>-big</i>	; assemble big endian (<i>-little</i> for little endian)
<i>-o</i>	; output object module <i>HDWCFG.O</i>
<i>>HDWCFG.ERR</i>	; redirect assembler error messages to <i>HDWCFG.ERR</i>

```
MWASM -big -o HDWCFG.O 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 Metrowerks version of the AMX library specification file *CJ382.LBM* as a guide.

Use the following command line switches when using the Metrowerks linker/librarian.

<i>-big</i>	; library is big endian (<i>-little</i> for little endian)
<i>-library</i>	; create a library
<i>-o YOURLIB.A</i>	; output library module <i>YOURLIB.A</i>
<i>>YOURLIB.LBE</i>	; redirect librarian error messages to <i>YOURLIB.LBE</i>

Make your library as follows.

```
MWLD -big -library -o YOURLIB.A @YOURLIB.LBM >YOURLIB.LBE
```

Metrowerks Processor Initialization Code

Metrowerks provides the following processor initialization modules, one of which must be compiled and linked with your AMX application.

```
...\PowerPC_EABI_Support\Runtime\Src\__ppc_eabi_init.cpp  
...\PowerPC_EABI_Support\Runtime\Src\__ppc_eabi_init.c
```

AMX was tested using a copy of the C++ module. This processor initialization file was renamed *PPCINIT.CPP* and compiled to generate object module *PPCINIT.O*.

Linking with the Metrowerks Linker

When used with Metrowerks C, the modules which form your AMX system must be linked in the following order.

```
PPCINIT.O          ; Processor initialization code
Your MAIN module
Other application modules

SYSCFG.O          ; AMX System Configuration Module
HDWCFG.O          ; AMX Target Configuration Module

CHxxxxxxT.O      ; AMX chip-specific clock driver or your equivalent
                  ; (not required if PowerPC decremter used as clock)

CJ382UC.O        ; AMX minimal C replacement library
                  ; (used to eliminate sprintf I/O support from C library)

CJ382UF.O        ; Launch and leave AMX (may be customized)

CJ382RAC.O       ; AMX ROM Access Module (customized)
                  ; (only if AMX placed in a separate ROM)
                  ; (see Appendix C in AMX PPC32 Target Guide)

CJ382CV.A       ; AMX PPC32 vc Conversion Library
                  ; (only if converting an AMX 86 v3, AMX 386 v1 or
                  ;      AMX 68000 v2 application)

CJ382.A         ; AMX PPC32 Library
                  ; Metrowerks C Runtime Libraries are automatically loaded
                  ; per environment variable MWLibraryFiles
```

Note: If you are using little endian hardware, you must create and use little endian versions of the Metrowerks processor initialization module `__ppc_eabi_init.c` (or `.cpp`) and the relevant runtime libraries.

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

Create a linker command file *YOURLINK.LCF*. Use the Metrowerks version of the AMX Sample Program linker command file *CJSAMPLE.LCF* as a guide.

Start with the sample link specification file and linker command 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 Metrowerks linker and locator using the following command line switches.

```
-big          ; link big endian (-little for little endian)
-unused      ; include unused symbols in the map file
-g          ; (optional) generate full debug info
-o          ; direct link output to file YOURLINK.OUT
-srec       ; direct HEX output to file YOURLINK.HEX
            ; generate Motorola S-record format
-map        ; direct section and symbol summary to file YOURLINK.MAP
-lcf        ; use linker command file YOURLINK.LCF
@          ; use link specification file YOURLINK.LKS
>          ; direct link error messages to file YOURLINK.LKE
```

The link and locate command line is therefore of the form:

```
MWLD -big -unused -o YOURLINK.OUT -srec YOURLINK.HEX
      -map YOURLINK.MAP -lcf YOURLINK.LCF
      @YOURLINK.LKS >YOURLINK.LKE
```

The resulting load module *YOURLINK.OUT* is suitable for use with the Metrowerks CodeWarrior PowerPC debugger.

The resulting load module *YOURLINK.HEX* is ready for burning into EPROM.

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 *CJ382ROP.S*, ROM Access Module *CJ382RAC.S* and ROM Option linker command file *CJ382ROP.LCF*.

The AMX Configuration Manager must have access to the ROM Option Linker Command Template file *CJ382ROP.LCT*. If you have installed AMX for multiple toolsets, the Manager may not be referencing the Metrowerks toolset directory *TOOLME* for its template files. Go to the **File, Templates...** menu and, from the list of selectors, choose the selector for the ROM Option Link/Locate File. Adjust the configuration template by browsing for the file *TOOLME\CFG\CJ382ROP.LCT*.

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

```
MWASM -big -o CJ382ROP.O CJ382ROP.S >CJ382ROP.ERR
```

```
MWASM -big -o CJ382RAC.O CJ382RAC.S >CJ382RAC.ERR
```

The AMX ROM is linked using linker command file *CJ382ROP.LCF* and link specification file *CJ382ROP.LKS* as follows.

```
MWLD -big -unused -m cjksender -o AMXROM.OUT  
-srec AMXROM.HEX -map AMXROM.MAP  
-lcf CJ382ROP.LCF @CJ382ROP.LKS >AMXROM.LKE
```

This example generates file *AMXROM.HEX* in Motorola S-record format suitable for transfer to ROM.

Note that command line switch *-m cjksender* is used to prevent loading of the default C startup module.

When you link your AMX application, be sure to include your customized AMX ROM Access Module *CJ382RAC.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 *CJ382ROP.S*, ROM Access Module *CJ382RAC.S* and ROM Option linker command file *CJ382ROP.LCF*.

Copy the ROM Option and ROM Access template files *CJ382ROP.CT* and *CJ382RAC.CT* to the current directory. Also copy the ROM Option Linker Command Template file *CJ382ROP.LCT* to the current directory.

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

```
CJ382CG HDWCFG.UP CJ382ROP.CT CJ382ROP.S
CJ382CG HDWCFG.UP CJ382RAC.CT CJ382RAC.S
CJ382CG HDWCFG.UP CJ382ROP.LCT CJ382ROP.LCF
```

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

Metrowerks CodeWarrior Debugger

The Metrowerks CodeWarrior™ PowerPC Debugger supports source level debugging of your AMX PPC32 system.

The CodeWarrior Debugger can operate by using a BDM connection to the PowerPC.

The CodeWarrior Debugger can also operate using a serial (or other) connection to the target PowerPC system under test. When used in this fashion, you must install the CodeWarrior *MetroTRK* Target Resident Kernel in your target hardware. Instructions for doing so are provided in the CodeWarrior Reference Manual. Your version of the Target Resident Kernel must provide a device driver for the serial (or other) device used for communication with the CodeWarrior Debugger. It is recommended that your driver use polled I/O so that the Target Resident Kernel can operate with interrupts disabled.

Using the *KwikLook* Fault Finder

The *KwikLook*™ Fault Finder is compatible with the CodeWarrior 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 CodeWarrior share a common link to the target system.

This page left blank intentionally.