V M L A B S

The Hitchhiker's Guide to Merlin

An Overview of the VM Labs Development System

Revision 0.08
17-Nov-98

VM Labs, Inc.
520 San Antonio Rd
Mountain View, CA 94040

Tel: (650) 917-8050
Fax: (650) 917-8052
www.vmlabs.com
www.nuon-tech.com

Note: This document is a work in progress. Please address comments or report errors to Mike Fulton at VM Labs (mfulton@vmlabs.com).

Table of Contents

This page intentionally left blank

1. Introduction

1.1 Unpacking The Development System

Congratulations on becoming a Merlin developer. You've just received your development system and have opened up the box. What's next?

After opening the box, you may look through the box and come to the conclusion that some items are missing. Where are the cables and controllers and other items that should go along with the system?

Don't worry, because it's not likely anything is missing. The controllers, cables, and other "missing" items are normally packaged inside the case of the development system. You must open the case and remove these items before applying power to the unit.

The table below allows you to check off the items which should be included in the package:

Item	Check
Cable: AC Power	
Cable: Controller Adapter Box to MDS	
Cable: Null modem	
Controller	
Controller Adapter Box	
Documentation Package	
Merlin Development System (MDS)	

Chapter 1 will introduce you to some details about VM Labs Developer Support, such as how to contact support personnel, how to log onto the support FTP site, and how to download the latest versions of the SDK and other files.

Beginning with chapter 1, we'll walk you through the steps of getting your system put together and connecting it to your host PC.

Later chapters will introduce the various tools of the SDK, demonstrate how to build and execute a sample program, and help you with trouble-shooting.

This page intentionally left blank

2. VM Labs Developer Support

2.1 Contacting Developer Support

You may contact the VM Labs Developer support staff via telephone, fax, E-Mail, or regular mail as follows:

VM Labs Developer Support	
E-Mail:	devsupport@vmlabs.com
Phone:	+1 (650) 917-8050 (see note below)
Fax:	+1 (650) 917-6610 (see note below)
Mailing Address Before Oct. 1, 1998:	VM Labs Attention: Third Party Developer Support 167 S. San Antonio Rd. #17 Los Altos, CA 94022
Mailing Address After Oct. 1, 1998:	VM Labs Attention: Third Party Developer Support 520 San Antonio Rd. Mountain View, CA 94040

Our support engineers may also be contacted individually:

Mike Fulton	
E-Mail	mfulton@vmlabs.com
Phone	+1 (650) 947-8270 (see note below)
Pradip Fatehpuria	
E-Mail	pradipf@vmlabs.com
Phone	+1 (650) 947-8271 (see note below)

NOTE: *VM Labs is planning to move into new offices during September/October 1998. Both the new and old addresses are shown above. Note also that this move may also require new telephone numbers. If the old numbers don't work, then please send EMAIL, contact directory assistance, or look on the VM Labs website at* ***http://www.vmlabs.com****.*

2.1.1 Sending E-Mail to Developer Support

The preferred method for developers to contact developer support is via E-Mail. In order to help ensure a swift and accurate response, please always try to provide the following information in your message:

- Your name

- Your company
- The name of the project you are working on.
- Your return E-Mail address
- Your telephone number. In some cases, our support engineers may decide that calling you would be more efficient and effective than an E-Mail reply.
- As much detail as possible about the problem you are having or the question you want answered.

2.1.2 Response Time

When you send an E-Mail message or fax, our support staff will usually be able to respond within 24-hours, except across weekends and holidays. Please keep in mind that certain special circumstances may sometimes affect response time.

While we do our best to avoid it, things do occasionally slip through the cracks. If you do not receive at least an acknowledgement of your inquiry within a few business days, please try again.

2.2 Contacting Non-Support Personnel

Occasionally, to help resolve a problem, our support engineers may put you into direct contact with another engineer who is not part of the regular support team.

In such cases, please keep in mind that non-support personnel are normally quite busy with their normal everyday assignments. Unlike the support staff, they are not accustomed to dealing with outside developers on a regular basis and cannot always devote their complete attention to your problem. They will do their best, but may not be able to respond as quickly as you would like.

When you contact non-support staff regarding an issue, please do your best to keep the original support staff member in the communications loop. Please make sure the support staff receives copies of any correspondence. This will ensure that any problems or confusion is kept to a minimum.

2.3 Developer Support Web Site

The VM Labs website is located at:

```
http://www.vmlabs.com
```

At the time of this writing, the developer support web pages are still in the planning stages. Refer to the main VM Labs web pages for the address and details about availability and signup information.

2.4 Developer Support FTP Site

VM Labs maintains a secure FTP site for 3rd party developers to download updates for tools, libraries, sample code, and other information. It can also be used for developers to send files to VM Labs in secure fashion. The URL is:

```
ftp://204.31.130.4
```

Your user ID and password for the FTP site are provided to you separately. If you do not have this information, please contact VM Labs developer support.

2.4.1 Downloading Files From the FTP Site

Always configure your FTP program for a BINARY transfer when downloading files from the FTP site. With many FTP programs, the default transfer mode is ASCII, and this may cause changes to the file during transfer.

If you don't have a separate FTP program, don't forget that Windows 95/98/NT comes with a simple console-based FTP client program called "FTP". It's not pretty, but it will serve the purpose. The table below lists the main commands you'll need.

Windows 95 FTP Program Basic Commands	
ASCII	Switch to ASCII download mode
BINARY	Switch to BINARY download mode
CD <directory>	Changes to the specified <directory>
DIR <specification>	Lists files matching the given <specification>
GET <file>	Downloads the specified <file> to the current local directory
HELP [command]	By itself, lists available commands. If a particular command is specified (i.e. "help get") then more detailed help on that command is shown.
PUT <file>	Uploads the specified <file> (by default in the current local directory) to the current directory on the FTP site.

2.4.2 Obtaining The Merlin SDK

The Merlin SDK is available to authorized 3rd party developers for downloading from the FTP Site. The file containing the SDK tools, demos, and sample code is:

```
SDK/SDK_0709.DES
```

The "0709" portion of the filename means July 9th and is subject to change. Later revisions will have filenames that indicate later dates.

The "DES" filename extension indicates that the file is DES-encrypted. You may decrypt it using the DES utility, which is also available on the FTP site at:

```
DES\DES.EXE
```

2.4.2.1 Decrypting the SDK File

The command line used to decrypt the SDK file is:

```
des -3Dk SecretPassword sdk_0709.des sdk.zip
```

where *SecretPassword* must be replaced by the real password, which is provided to you separately along with your FTP login information.

Please note that the command line options and password for the DES utility are case-sensitive. The command line above will create **SDK.ZIP** which can then be used with any standard Windows-95 ZIP file tool.

We're not going to provide details about installing the SDK just yet, because there are other important issues which we should discuss first. However, if you want to peek ahead, see section 6.2, "***Tools Installation***".

2.4.3 Obtaining Other Files

Please note that there may be other files containing sample code and/or demos available from the FTP site. For example, the SDK/DEMO0709.DES file contains a number of demo programs (mostly without source code) which are non-essential, but which may be interesting.

2.4.3.1 Decrypting Other Files

Please note that all files on the FTP site posted on the same date will use the same decryption password. For example, SDK_0709.DES and DEMO0709.DES share

the same password. Therefore, simply apply the instructions given in section 2.4.2.1 above, using different filenames on the command-line as appropriate.

2.4.3.2 Personalized Downloads

Please note that from time to time, the DOWNLOAD directory on the FTP site may have files intended for specific developers. For example, a developer may send some source code to a developer support engineer who will make changes and post a revised version for the developer to download.

Such personalized downloads will always be encrypted with a unique password to ensure confidentiality. If you aren't the intended recipient, then you will not be able to decrypt these files.

2.4.4 Obtaining Technical Notes

The VM Labs Support FTP Site contains a variety of technical notes that may be useful and interesting. Please look in the *TechNotes* directory on a regular basis for updates and new materials.

2.4.5 Uploading to the FTP site

On occasion, you may wish to upload files to the VM Labs FTP site.

Log onto the FTP site in the usual fashion, then change to the "/upload" directory. You won't be able to get a file listing in this directory, but it is the only place on the FTP site where you can upload a file.

Please make sure your upload is accompanied by a telephone call or EMAIL message to the person at VM Labs intended to receive the file.

2.5 Things to Keep in Mind About Developer Support

There are a number of things about the whole developer support process that are important to know and remember. When contacting developer support, please keep them in mind. This will make the entire developer support process much smoother for everybody.

2.5.1.1 "We are both under non-disclosure"

Sometimes developers are reluctant to share information or code fragments that are required to find the solution to a particular problem. This might be because your project hasn't been publicly announced, or for many other reasons.

Please always keep in mind that the relationship between VM Labs and a developer is covered under a mutual non-disclosure agreement. Our support staff is not going to discuss your confidential information with the press, the public, or even other people within VM Labs who do not have reason to know about it.

If there's any doubt about which information is "confidential" and which might be considered "public", then just let us know which is which. For example, details about the inner workings of your code are always going to be "confidential" and there's no doubt about that.

On the other hand, just the fact that you're working on a particular project could be either "public" or "confidential" depending on a number of factors.

When in doubt, let us know in advance so that we can take appropriate care. That way, there should be no reason for you to be reluctant to share information about your project with our developer support staff.

2.5.1.2 "I know it sounds stupid, but please try it anyway?"

Frequently a developer support engineer will make a suggestion that sounds very simplistic and unlikely to solve the problem. If the suggestion is more than minimally time consuming, the developer may be reluctant to try it.

It's amazing how often those "stupid" suggestions lead directly or indirectly to the solution.

Experience shows that a very high percentage of developer support issues are based on very simple problems or mistakes that are easy to overlook. Finding the cause of a problem is often a matter of collecting as much data as possible. Even "stupid" suggestions will help eliminate possibilities and the results often provide data that will help determine the real problem.

2.5.1.3 RTFM: Read the fine manual

That is what "RTFM" stands for, you know. At least, that's how we'll define it here and now.

We've all heard "RTFM" in response to a problem. It's a cliché, but it got to be one because it's true so often. A surprising number of times, the answer to your question is right in front of you, someplace within the documentation at hand.

We acknowledge that finding a specific piece of information in a sea of documentation can sometimes be difficult, so please don't hesitate to contact developer support whenever you feel it's necessary. But always keep in mind that if the answers are in the documentation, you'll find them faster yourself than if you contact developer support.

It's difficult and impractical to commit large amounts of technical documentation entirely to memory. However, it's really helpful if you make an effort to at least scan through all of the documentation so you have a basic idea of what subjects are covered, and in which section of the docs they are located. This will help you more quickly locate the information you need when you need it.

2.5.1.4 "The support guy's asking me too many questions, instead of answering mine!"

Sometimes when a developer asks some questions, instead of giving the answer they wanted, the support engineer asks questions: What they are doing? And why are they doing it that way? The developer then gets annoyed at being grilled for details, and often suspects one or more of the following:

- They just don't want to give me that information. Maybe it's "undocumented".
- They think I'm doing something "illegal" and they want to catch me at it.
- They don't think I know what I'm doing.

The result is a situation full of distrust and frustration on both sides, which doesn't do anybody any good.

In most cases, the reason for the support engineer's questions is that providing the best answer to the developer's original question depends greatly on the context of the situation, and therefore having more detail is necessary to provide an answer.

Maybe the original question is not be very detailed, and more detail is needed in order to distinguish one problem from another. Or maybe the description of the problem is unusual in some fashion, and more detail is needed in order for things to make sense.

The main goal of a developer support engineer is always to help developers past their problems. There are two components involved in accomplishing this goal: first, figuring out problems that pop up for the first time; second, giving out solutions to problems that have previously hit other developers.

Whenever a problem comes up for the first time, the developer support engineer has to try to help solve the problem, but must also prepare for the possibility that the same problem will eventually happen to someone else. Different people might describe a single problem in drastically different ways. Without details, it can be difficult to determine that they are talking about the same basic problem.

That's why we ask so many damn questions. ☺

Having said all that, we'll also say that it's true that there may be rare occasions when we may suspect the developer is doing something "illegal" or is trying to access something which is undocumented. Such occasions are very much in the minority, but they do occur once in a while.

Please keep in mind that we're not trying to catch a developer doing something wrong in order to slap them on the wrist or to berate them for not following the rules. Our goal is to make sure the developer's software will operate properly and efficiently, and to avoid compatibility problems with different pieces of hardware.

3. Connecting Your Merlin

The back of your Merlin development system contains a variety of connectors, as shown in the diagram below:

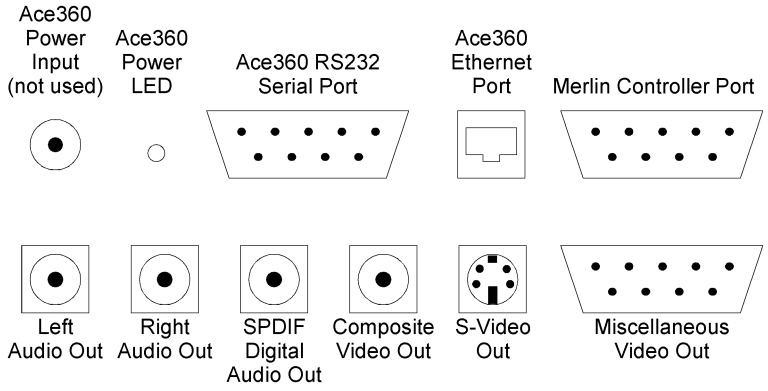

Figure 1 — Merlin Development System Ports

The ports along the top row are used to connect your Merlin to your host computer system, and to connect a game controller. The ports along the bottom row are used to connect your Merlin system to a television, monitor, and/or stereo system.

As we go through this section, we'll discuss what gets connected to each of these ports.

3.1 Connecting Merlin to your TV or monitor

The Merlin system uses standard RCA cables for audio and composite video outputs. If your TV or monitor supports it, you may use the S-Video output for improved video quality. These cables may be obtained at most audio/video electronics dealers.

The Merlin development system does not include RF modulated output for connection to a television's antenna leads. Your television or monitor must have composite video or S-Video inputs.

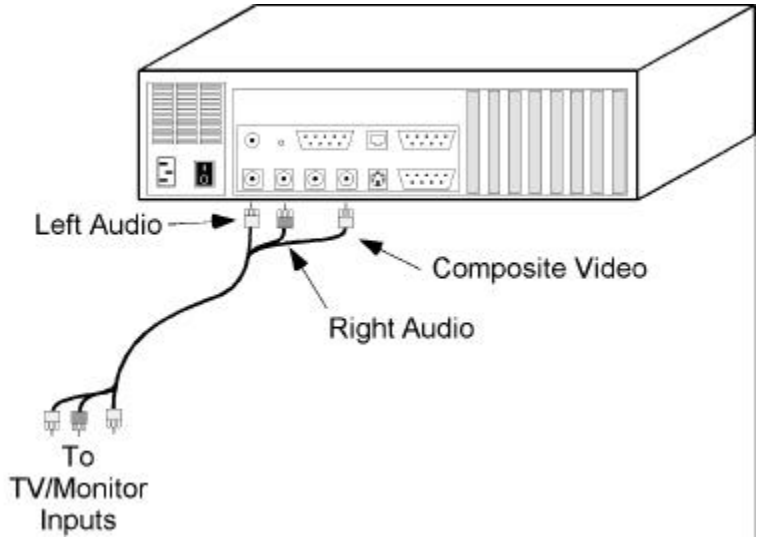

Figure 2 — Standard Video & Audio Connections

Warning: Do not connect the SPDIF Digital Audio output on the Merlin to an analog audio input on your television, stereo, or other device. Doing so may result in speaker damage. Connect this output signal only to a coaxial SPDIF digital audio input.

3.1.1 NTSC & PAL

Merlin's video display system is configured by the system BIOS whenever the system is powered-up or reset. Currently, the display is always set for standard NTSC video output with a 60 Hz refresh rate.

The ability to configure the video display for PAL mode with a 50 Hz refresh rate is not available at the time of this writing, but this may have changed by the time you read this. Please be alert for updates and if necessary, contact VM Labs Developer Support for more information.

3.1.2 Building your own video cable

You may wish to build your own cable to connect the Merlin Development System to a monitor without standard composite or S-Video inputs (for example, a SCART monitor). Here are the pin-outs of the 9-pin ***Miscellaneous Video Out*** connector:

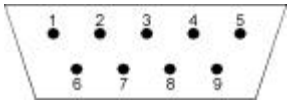

Figure 3 — Miscellaneous Video Out Connector

Pin:	Signal:
1	Composite video (used as sync for RGB SCART)
2	Composite sync (TTL levels, for Amiga-type monitors)
3	Blue
4	Green
5	Red
6	Ground
7	+12V
8	+3.3V
9	Ground

3.2 Connecting the Merlin Controller

In the currently shipping version of the development system, the game controller does not attach directly to Merlin. Instead, we are using a controller adapter box that goes between the controller and Merlin. The adapter is a small black plastic box with a female DB-25 connector on one side and a male DB-9 connector on the other. You should also have a cable with female DB-9 connectors at each end (maybe with a gender changer at one end).

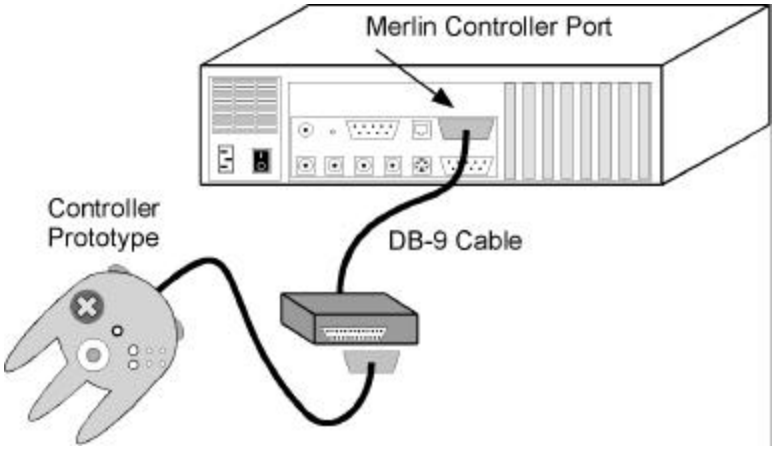

Figure 4 — Connecting the Merlin Controller

Connect the DB-9 cable between the Merlin Controller Port and the small black box. Plug the controller the other side of the small black box. That's it!

3.3 Connecting Merlin to your Host Computer

The Merlin development system is designed to connect to your host computer via standard 10Base-T Ethernet running on a standard twisted pair RJ45 cable. Category 5 or better cable is recommended.

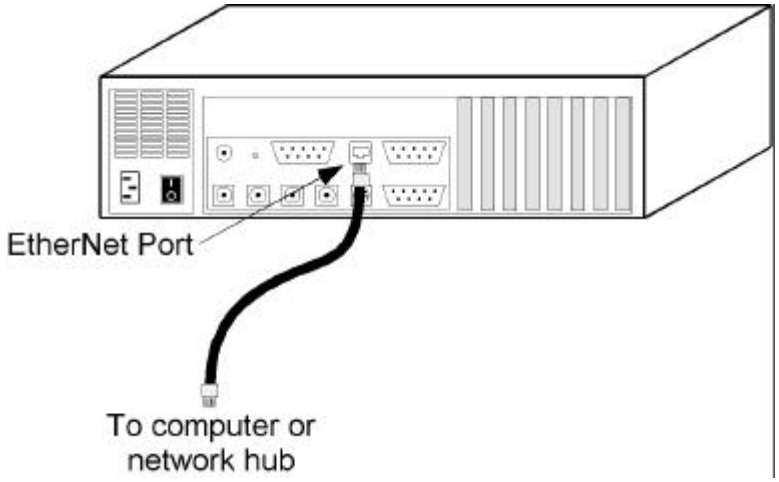

Figure 5 — Merlin Ethernet Connection

3.3.1 Connecting as part of an existing network

If your host computer is already part of a network, simply connect a standard twisted pair RJ45 network cable between the Merlin's Ethernet port and your network hub.

You may need to obtain a different network hub if your current one does not support 10Base-T on twisted pair RJ45 cable. If your network uses BNC coaxial cables, the best solution is to find a small hub that supports both twisted pair RJ45 and BNC. The computer can be connected using the existing cables, but the Merlin must be connected using a twisted pair RJ45 cable.

If your network is running 100Base-T, the best solution is to use a hub which allows both 100Base-T and 10Base-T connections.

3.3.2 Connecting directly

If your host computer is not connected to a network, then it may instead be connected directly to the Merlin development system using a special *crossover*

network cable. This cable should run directly from the Merlin's Ethernet port to a network adapter card installed in your host computer.

If your host computer does not have a 10Base-T Ethernet adapter using twisted pair RJ45 cable, you will need to install one.

Note: A crossover network cable is colored bright orange to distinguish it from standard cables. Making your own crossover cable is not recommended.

3.3.3 Merlin ACE360 Serial Port

The Merlin system also has the Ace360 serial port. However, because the transfer rate of an RS-232 serial port is relatively very slow compared to Ethernet, it is used only for configuration of the Merlin system and other special purposes, and not for general communication.

This page intentionally left blank.

4. Communicating With Merlin

The TCP/IP network communications protocol is used to communicate with the Merlin development system. Before you can run sample programs or do anything else with Merlin, you must make sure that the TCP/IP setups for Merlin and your host computer are correctly configured.

4.1 Do I have TCP/IP?

If your computer is already connected to a network, and you have the ability to access the Internet via that network, then you already have TCP/IP installed. You will probably not need to change anything about your host computer's network configuration.

If you access the Internet only via modem, or do not have Internet access at all, then you may need to enable and/or configure the TCP/IP protocol so that it may be used with your network adapter to communicate with the Merlin development system.

4.2 Network Configuration

Merlin requires no special network configuration for your host PC beyond enabling the TCP/IP protocol.

When configuring network hardware and software, if you do not have a strong working knowledge of the TCP/IP protocol and network configuration, it is recommended that you contact your in-house network administrator for assistance.

4.3 Merlin's IP Address

The IP address of the Merlin development system is set by default to 192.1.1.*xxx*, where *xxx* is indicated by a sticker on the back of the system.

Please note that the Merlin development system currently requires that the host PC and Merlin to be on the same subnet in order to communicate. This means that the first three values of the IP address must be the same on both machines. For example, if your host computer's machine is set to 203.34.123.54, then Merlin must be set to 203.34.123.*xxx*, where *xxx* must be a value from 0-255 which is not used by the host computer or any other machine on the same subnet of your network.

Remember that the task of assigning IP addresses is usually the responsibility of your in-house network administrator. In order to avoid conflicts with other network devices, always ask them to assign an IP address for Merlin to use.

4.3.1 Changing Merlin's IP Address

Changing the IP address used by the Merlin's ACE360 board is done in one of two ways, depending on which version of the system boot ROM is installed in your system.

Newer versions of the system boot ROM include a menu driven configuration program which can be accessed at power-up or after a hardware system reset. If your system has this version of the boot ROM, then please see chapter 5 for additional information.

With older versions of the system boot ROM, following the steps below performs the IP address configuration:

1) Connect a DB-9 (female) null-modem serial communications cable from an available COM port on your host PC to the ACE360 serial port on the back of your Merlin box.

2) Run "HyperTerminal" (HyperTerminal is included with Windows 95/98, and is normally accessed via the *Accessories* group in your Windows 95 Start menu.

3) Create a new HyperTerminal session using the following parameters:

   Port = COMx (where 'x' indicates whichever port you're using)
   Baud = 9600
   Data bits = 8
   Parity = none
   Stop bits = 1
   Flow control = none

4) If it is powered on, turn off your Merlin system for a few seconds, then turn it back on. If you've successfully established a connection, then in HyperTerminal you'll see a startup message from Merlin that looks something like this:

```
Oz Debug Stub - Apr 07 1998, 16:24:06
TELNET_PROTOCOL defined
FNS device Quicc0 installed at 0x0
Listener initialized
```

```
DVD not Enabled!
Enter Ctl-C to go interactive.
```

If you see no text at all, or if the text is garbled, then you do not have a valid connection. You should check your null modem cable connections and your HyperTerminal settings.

Once you've established a connection, hit <Control-C> on your keyboard to bring up the ACE board's configuration menu. If you do not get this menu, check that your HyperTerminal settings match those specified in step 3 and then try again. If nothing works, contact VM Labs for assistance.

When the configuration menu comes up, select 'E' for EEPROM functions. For a list of available EEPROM functions, type '?' and press <Return>.

To see the current network IP address settings, type 'S' and press <Return>. It's a good idea to write down the existing network settings, before making changes.

You may change the IP address by typing "I" followed by the new IP address in dotted notation. For example:

```
I 209.46.69.224
```

would change the IP address to 209.46.69.224

To change the IP mask address, type "M" followed by the new IP mask address. For example:

```
M 255.255.255.0
```

would change the IP mask address to 255.255.255.0. (This setting will be the one used in almost all cases.)

When you've finished making your changes, type 'S' again to verify the new settings. Remember to write these down. You always use a label on the back of the machine to indicate the current settings.

When you've finished, type 'X' to exit the ACE board configuration menu. Then cycle the power to Merlin so that the updated settings will be recognized.

That's it! You've changed your Merlin's IP address. You can now end your HyperTerminal session.

This page intentionally left blank.

5. Merlin Boot ROM

This section will discuss the Merlin boot ROM.

5.1 Configuration

Early versions of the Merlin development system featured a relatively simple boot ROM program that simply cleared RAM and then displayed a set of color bars.

Later versions of the development system boot ROM include a more sophisticated configuration program that allows developers to control a variety of important details about how the machine will handle the startup process.

At the time of this writing, detailed information about the new version of the system boot ROM is not yet available. More on this will be added to a future version of this document..

5.2 Boot ROM Versions & Demo Programs

Please note that certain older demo programs for Merlin may not function properly with newer versions of the boot ROM. In most cases, this is due to the fact that the system video setup may be done differently than on earlier versions of the boot ROM.

This mainly affects older demos which are provided without source code, not sample programs which are provided along with source code as part of the SDK.

This page intentionally left blank.

6. Merlin Development Tools

This section will give a basic overview of the Merlin development tools: how to get them working, what files are involved, etc.

6.1 System Requirements

The system requirements for the development tools are as follows:

- 133Mhz or faster Pentium-based PC
- 10Base-T (RJ45) Ethernet adapter
- Windows 95
- 16mb RAM (minimum, 32mb or more recommended)
- 50mb free disk space (the basic amount required for the SDK tools, libraries, sample code, and demos, not counting your own project's code and data)

Please note that the tools may work with lesser configurations, but they are not recommended

6.1.1 Windows 98

The official requirements for the SDK tools is Windows 95. However, everything should work equally well under Windows 98. There are no known problems unique to Windows 98.

6.1.2 Windows NT

To our knowledge, the SDK tools should also work under Windows NT 3.5 or 4.0, but this has not been tested.

Since the SDK tools communicate with Merlin over TCP/IP, no special Merlin-specific hardware drivers are required.

Please note that a small change to the DJGPP.ENV file may be necessary if running under Windows NT. Please see section 2) below for more information.

6.2 Tools Installation

Information on obtaining the Merlin SDK is provided in section 2.4.2. We'll presume you've followed the instructions there and have decrypted the downloaded file to obtain an archive file named SDK.ZIP.

You should extract the contents of the SDK.ZIP archive to the root directory of your selected drive. When expanding the SDK archive, make sure you have selected the option to preserve the archive's directory hierarchy. This will result in a folder named VMLABS that will contain everything else.

Note that using the standard DOS-based PKZIP or PKUNZIP tools from a DOS command shell will not preserve the long filenames used by many of the files within the archive. A Windows 95/NT compatible ZIP tool that properly understands long filenames is required to properly extract the contents of the archive.

If you do not have such a tool already, please visit the DOWNLOAD.COM web site at **http://www.download.com** and search for "WINZIP".

After you have extracted the contents of the SDK archive to your hard disk, most of the Merlin development tools require no additional setup except for setting a few environment variables, as detailed in section 6.3 below.

6.3 Environment Variables

The environment variables listed in the table below must be set as indicated in order for the Merlin SDK tools to function properly:

Variable Name	Description
DJGPP	This must point to the configuration file used by the C/C++ compiler, normally the \VMLABS\CFG\DJGPP.ENV file, on the drive where you have installed the SDK.
MD_PORT	This must be set to the IP address assigned to the Merlin development system. For Windows 95/98, this is normally configured by a line in your AUTOEXEC.BAT file. If you have multiple Merlin development systems and wish to communicate with more than one, you must change the contents of the MD_PORT variable to switch back and forth.

Variable Name	Description
PATH	List of directories to search when looking for executable programs. For Windows 95/98, this is normally configured by a line in your AUTOEXEC.BAT file. The PATH list must include the VMLABS\BIN directory on the drive where you have installed the SDK. You can easily add this to your existing path using the following command: PATH=%PATH%;c:\vmlabs\bin This presumes that you've installed the SDK on drive C. if not, change the example as appropriate. If you have development tools for other platforms installed on your system, check to see if anything has a filename similar to any of the Merlin tools. If so, this is likely to cause problems. You may need to change the order of the directories specified in the PATH variable, or you may need to remove the directory containing the other variables from the PATH. You may wish to create batch files that reset the PATH variable to include or exclude the VMLABS\BIN directory as needed in order to switch back and forth to other tools.

6.4 The DJGPP.ENV File

The Merlin C/C++ compiler is based on the GNU GCC compiler.

Most versions of the various GNU tools for various computer platforms rely on a variety of environment variables to determine the disk directory where important files are located. However, MSDOS and MS Windows by default have a very small environment space for storing global environment variables. This meant that users had to expand the environment space so that enough room was available for all of the variables used by the GNU tools.

To make things easier to configure, the DJGPP versions of the GNU tools have standardized on the use of just one environment variable, "DJGPP", which specifies the path used to located the DJGPP.ENV file.

The DJGPP.ENV file contains all of the definitions for directory paths which were previously specified in environment variables.

The default DJGPP.ENV file is configured so that no changes are required, provided that the SDK is installed at the root directory of whatever drive you've chosen. This means your "VMLABS" directory should be located in the root, with the "BIN" and other directories one level deeper inside that one.

6.4.1 Changing DJDIR

Should you wish to install the Merlin SDK other than at the root directory, you will need to edit the DJGPP.ENV file to recognize the new location. The "DJDIR" variable must specify the complete name of the folder where the "BIN" directory is located.

For example, if you've installed the SDK into a folder named "D:\DEVELOP", then you'd follow the steps below:

1) Load the file into a text editor such as the Windows NOTEPAD program. A few lines down from the top of the file, you should see a line that looks like this:

```
DJDIR=%:/>DJGPP%/..
```

2) Change the line to look like this:

```
DJDIR=d:/develop/vmlabs
```

6.4.2 Using Windows NT

Windows NT is not currently an officially supported platform for the VM Labs SDK. However, should you wish to use that system, note that some changes in your setup may be required for things to work properly.

Under Windows NT, the current version of the Merlin SDK's C/C++ compiler may not deal with long filenames correctly. It may be necessary to change the name of certain header files or source code files in order to make them conform to the original 8.3 filename size limitations. Keep in mind that some header files may in turn include other header files, and may require editing to reflect any changes in filenames.

6.5 Making the tools work

The main things required to make the development tools work are:

1) Make sure your Merlin's IP address is configured properly. See section 4.3 for further information.

2) Download the most recent version of the SDK and decrypt it using the DES tool as described in section 2.4.2. This will give you a ZIP archive file containing the SDK files.

3) Extract the contents of the SDK ZIP file as described in section 6.2.

4) Set the environment variables used by the tools as described in section 6.3.

5) If necessary, edit the DJGPP.ENV file to reflect your installation, as described in section 6.4.

6.6 Files Used By The Development Tools

This section will break down the programs included in the SDK into separate groups according to function. This is primarily a reference so that you know what a particular file is for.

Please note that some files may be listed in multiple places. Also, some tools may not be listed because they have been removed from the SDK. This is usually because the tool in question has been replaced by improved functionality in another tool, or because it was never really intended to be part of the SDK in the first place.

6.6.1 C/C++ Compiler

There are several files that make up the C compiler:

Program Name	Description
AS.EXE	Merlin LLAMA assembler. Assembles the intermediate assembly language output created by the C/C++ compiler.
CC1.EXE	C Compiler executable
CC1PLUS.EXE	C++ compiler executable
CPP.EXE	C/C++ Preprocessor
GMAKE.EXE	Program builder utility

Program Name	Description
LD.EXE	Linker used to combine compiled object modules and libraries into an executable program file.
MG++.EXE	C++ compiler driver
MGCC.EXE	C compiler driver

6.6.2 Assembler

The main assembler is known as Llama. The AS assembler is another incarnation of Llama which is intended to be called as the assembler stage of the C/C++ compiler.

Program Name	Description
AS	Same thing as LLAMA. Assembler stage of the GCC compiler.
LLAMA	Merlin assembler

6.6.3 Linker

The main linker is VMLD. The LD linker is another incarnation intended to be called as the linker stage of the C/C++ compiler.

Program Name	Description
LD	Linker used to combine compiled object modules and libraries into an executable program file.
VMLD	Same thing as LD. VMLD is normally used when calling the linker directly, while LD is used by the C compiler.

6.6.4 Debugger

At the time this was written, it was expected that a Merlin version of the GNU GDB C/C++ source-level debugger would be available soon.

Program Name	Description
PUFFIN	Console version of debugger
PUFFINW	Windows version of debugger

6.6.5 Object Module Utilities

The following tools are designed to perform various operations on compiled object module files or executable program files.

Program Name	Description
COFF2MPO	Converts a COFF object module into the MPO format. This program is obsolete and will be removed from the SDK because COFF loading now built into MLOAD.
COFFDUMP	Dumps a list of all symbols within a COFF object module or executable program file. This program is obsolete and will be removed from the SDK because VMNM is now available.
MPO2COFF	Converts an MPO object module into COFF format. This program is obsolete and will be removed from the SDK because COFF loading now built into MLOAD.
VMAR	Library archive utility
VMNM	Library & object module symbol name utility
VMOCOPY	Object module manipulation & conversion utility
VMSTRIP	Symbol strip utility.

6.6.6 Miscellaneous Tools

The following tools are used for a variety of purposes.

Program Name	Description
MLOAD	Aside from the debugger, this program is the programmer's main interface to Merlin. It allows you to download code and data, reset the machine, update the flash ROM, and more.
REDIR	Allows user to redirect the 'stdout' and 'stderr' character devices used by the tools for error message, status information, etc.

This page intentionally left blank.

7. Tools Mini Command Reference

This section will provide a brief description of the main command line options for the primary SDK tools, and also provide other basic information which may be useful.

This is intended to serve primarily as an introduction. Some tools will feature more command line options and features not mentioned here. For more detailed information on any given tool, please also refer to whatever additional documentation is provided separately.

7.1 C/C++ Compiler

The program normally used to execute the C/C++ compiler is MGCC. It uses a command line formatted as follows:

```
mgcc [options] [source files]
```

The table below shows some of the more useful command line options for MGCC. Please note that the commands are case-sensitive.

Option	Description
-ansi	Force strict ANSI-C syntax. Disables GCC features which are not ANSI-compliant, including the `asm` and `inline` keywords, certain predefined macros, and recognition of C++ style comments in C code.
-c	Perform a compile operation only, do not call the linker
-C	Tells the preprocessor not to discard comments. Used in conjunction with the "-E" option.
-D<*macro*>[=*value*]	Defines a preprocessor *macro* on the command line. This is equivalent to having "#define *macro*" statements embedded at the top of the C source code. For example, "-DSCRNWIDTH=360" is equivalent to having "#define SCRNWIDTH 360" at the top of your source code file. The value field is optional. If no value is specified, the macro is assigned a value of "1". For example, having "-DNOJOYSTICK would be equivalent to having either "#define NOJOYSTICK" or "#define NOJOYSTICK 1".

Option	Description
-E	Stop compiling after the preprocessor stage is finished. If no output filename is specified using the "-o" option, the output from the preprocessor is sent to standard output.
-g	Add source-level debugging information to the output file.
-I *directory*	Add the specified *directory* to the paths searched for included files during the preprocessor stage.
-include *file*	Process *file* as input immediately before processing the source file.
-L *directory*	Add the specified *directory* to the paths searched for library archive files during the link stage.
-l<libraryname>	Specify that a particular library archive should be included in the link process. For example, "-lmath" would be used to include "math.a". The library specified is expected to be in the current directory, or in the standard system library directory specified within the DJGPP.ENV file. Libraries and object modules (including those resulting from source files) are always searched in the order specified.
-mreopt	Invoke the assembler with -b -O1
-mreopt-more	Invoke the assembler with -b -O2
-no-builtin	Don't recognize built-in functions whose names do not begin with two leading underscores. This includes abort(), abs(), alloca(), cos(), exit(), fabs(), ffs(), labs(), memcmp(), memcpy(), sin(), sqrt(), strcmp(), strcpy(), and strlen(). GCC normally generates special code to handle certain built-in functions more efficiently. However, this can affect debugging, and you cannot change the behavior of those functions by writing customized versions.
-o <file>	Specify the filename used for output
-O0 -O -O1 -O2 -O3	Specify that optimization should be used, in varying degrees. "–O0" is no optimization. "-O" is basic optimization. "-O3" is extreme optimization. Note that this does not specify optimization for the assembler stage. To specify assembler optimization, see the **mreopt** and **mreopt-more** commands.

Option	Description
-pedantic	Issue all warnings required by strict ANSI standard C. Reject forbidden extensions.
-S	Compile only, do not call the assembler, do not delete the assembly language source file that gets generated
-s	Remove all symbol table and relocation information from output file.
-traditional	Support some aspects of older non ANSI-C compilers. May not work with header files written to the ANSI-C specification.
-U *macro*	Undefine the specified *macro* symbol. Equivalent to "#undef *macro".* This option is always processed after the "-D" option and before the "-include" option.
-u *symbol*	Pretend that *symbol* is undefined, forcing linking of library modules in order to define it.
-v	Verbose mode. Output information about command lines to programs called by MGCC, such as the C preprocessor, assembler, or linker.
-Wall	Turn all warnings on.
-Xlinker *option*	Pass *option* through to linker stage. Note: if linker option requires arguments, the "-Xlinker" command must be given for each. For example, to pass through "assert definitions" you must write "-Xlinker assert –Xlinker definitions"

For example:

```
mgcc -O2 -mreopt-more -o hello.cof hello.c
```

This would specify extreme optimization for the C compiler, as well as optimization for the assembler. It specifies that the output filename should be HELLO.COF. Finally, the input file HELLO.C is specified. This will compile the HELLO.C file, link it with the standard libraries, and produce the HELLO.COF output file.

7.1.1 Function Calling Conventions

Generally, the first six words of parameters (counting from the left) are passed in registers r0 through r5.

Any excess parameters are passed on the stack, with scalar alignment.

Parameters with short or char type will be promoted to int (or unsigned int) before the call.

- A parameter will be placed on the stack:
    - If it is an unnamed parameter to a <stdarg.h> function, or
    - If it is the last named parameter to a <stdarg.h> function, or
    - If the type has variable size, or
    - If the type is marked as addressable (it is required to be constructed into the stack), or
    - If the padding and mode of the type is such that a copy into a register would put it into the wrong part of the register.
    - If it is too large to fit in the remaining parameter registers. In this case, subsequent parameters will still be candidates to be passed in registers.

A value in a register is implicitly padded at the most significant end. On a big-endian machine, that is the lower end in memory. So a value padded in memory at the upper end can't go in a register.

Once a parameter has been forced onto the stack, all the remaining parameters will go there too. (except as noted)

Return values are in r0. Five to eight byte return values are in r0 and r1. Bigger return values are in memory, with the address in r0.

The called function is responsible for preserving r12 - r31 and *acshift*. It is not responsible for preserving r0 - r11.

7.1.2 C/C++ Compiler Notes

- The compiler generates code that uses r31 as its stack pointer. It expects *acshift* to be zero. These get set by the code in the C runtime startup file CRT0.O.
- By default, type char is signed.
- Under some circumstances the compiler will use a push instruction to free up a vector. Be sure the hardware stack pointer always points someplace safe.

7.1.3 Additional Documentation

The C/C++ compiler in the Merlin SDK is based on the GNU GCC compiler from the Free Software Foundation.

Additional documentation on the GCC compiler, not specific to Merlin, is available in a separate document.

7.2 Llama Assembler

The Llama assembler was created by VM Labs to meet the specific requirements of the Merlin processor. It uses a command line formatted as follows:

```
llama [options] [source file]
```

The table below shows some of the more useful command line options for the Llama assembler. Please note that the commands are case-sensitive.

Option	Description
-?	Help. Display command line options.
-b	Assume condition codes need not be preserved across branches
-B *linkbase*	Set base address for linking (Valid for MPO output only)
-c#	Add padding for executing from cache; # is the length of cache lines in bytes. Using -c alone is the same as -c32.
-Dsymbol[=val]	Define a symbol and optionally assign it a value. Equivalent to: Symbol = value At the top of your assembly source code file.
-e *errfile*	Output XLisp compatible error records to *errfile*

Option	Description
-fasm [,bin] [,expand-syms] [,expand-includes] [,expand-all]:	Create assembly language output; options available: *bin*: annotate output with hex representation of instructions *expand-syms*: expand symbols to their ultimate definitions *expand-includes*: expand contents of .include directives *expand-all*: expand symbols and interpret module definitions
-fbinary:	Output raw binary data
-fcoff:	Output COFF object file
-f*FMT*	Select format of output file
-flist:	Same as **-fasm,bin,expand-includes**
-fm68k:	Same as **-fveri,width=32,segheader,prefix=' .dc.l 0x'**
-fmpo:	Output .mpo file for debugger / Merlin emulator
-fsrec:	Output Motorola S-Records
-fveri [,width=nn] [,segheader] [,absaddr] [,prefix =*'string'*] [,rom]:	Create Verilog load file; options available are: width=nn: set width of output file in bits (default 128) segheader: output 2 long word header for each segment: the segment origin and segment length in bytes absaddr: output absolute addresses in the Verilog file; otherwise the top bits of the address will be masked off and it will be divided by the memory width in bytes prefix='string': causes the given *string* to be printed at the beginning of each line, instead of a tab. Must be the last option given. rom: same as `width=8,segheader,absaddr'
-g	Include GDB debugging information
-i *incfile*	Process contents of file, but do not include it in assembly output

Option	Description
-I *incpath*	Add incpath to the search path for include files
-jextern	Default is –jlocal, unless -c was given
-jlocal	Assume jumps/jsrs are in local RAM
-nolines	Remove all line number info methods
-nolisp	Assume jumps/jsrs are in external (non-local) RAM remove all before and after methods
-o outfile	Set output file name
-O#	Optimization level: 0 = no optimization 1 = fast (but not very good) optimization n>1 = good optimization with n-1 levels of lookahead (potentially unbelievably slow) Default is -O0
-Osize	Optimize for space rather than time
-r#	Assembly language revision number. Default is: 20
-v	Verbose flag: print interesting statistics about the program and the file being assembled.

7.2.1 Additional Documentation

Additional documentation regarding the Llama assembler is available in two documents:

- ***Llama User's Manual***
- ***Optimizing Your Llama***

7.3 Linker

The linker from the Merlin SDK was created by VM Labs to meet the specific requirements of the Merlin processor. It uses a command line formatted as follows:

```
ld [options] [source files] [library files]
```

The table below shows some of the more useful command line options for the linker. Please note that the commands are case-sensitive.

Option	Description
-?	Help. Display command line options

Option	Description
-b *file*	Link in the specified *file* as raw binary information.
-B *value*	Use alternate base memory location. Default base is 0x80000000.
-e *symbol*	Define an entry point. Required in order to create an executable program file.
-I	Incremental link. Makes the linker produce a relocatable output file that can be used as input to another link.
-L*directory*	Add the specified *directory* to the path list searched for library files. Normally, the path list is specified by the VMLD_LIBRARY_PATH environment variable.
-l*name*	Include the library archive specified by *name* in the link. The prefix of "lib" and a filename extension of ".a" is automatically added to *name*. For example, "-lmath" specifies that the library archive LIBMATH.A should be included in the link.
-n	Generate an output file even if non-fatal errors occur.
-o *name*	Specify that *name* should be used for the output file. The default output filename is LD.OUT.
-T *name=value*	Assign an origin address of *value* for the section with the specified *name*. If that section does not exist, it will be created.
-V	Print version of linker

7.4 GMAKE Utility

The GMAKE utility uses a command line formatted as follows:

```
gmake [options] [target(s)]
```

The table below shows some of the more useful command line options for the GMAKE utility. Please note that the commands are case-sensitive.

Option		Description
-C *directory* --directory = *directory*	**or**	Change to the specified *directory* before doing anything
-d -debug	**or**	Debug mode. Print out MAKEFILE debugging information. Prints commands being executed, and lots of other stuff.

Option	Description
-e **or** --environment-overrides	System environment variables override variables with same name which are defined by GMAKE and within MAKEFILE. (Default is vice-versa.)
-f *file* **or** --file=*file* **or** --makefile=*file*	Specify that *file* is the makefile to be processed. If this option is not used, then GMAKE looks for a file named MAKEFILE in the current directory.
-h **or** -help	Print out command line options
-I *DIRECTORY* **or** --include-dir=*DIRECTORY*	Search *DIRECTORY* for included makefiles.
-I **or** --ignore-errors	Ignore errors. Continue with MAKE process even when commands return error.
-j *num* **or** -jobs *num*	Allow a maximum of *num* jobs at once. Default is unlimited.
-k **or** --keep-going	Continue with MAKE process even if some targets cannot be made.
-l *load* **or** --load-average *load*	Specify maximum load allowed.
-n **or** --just-print **or** --dry-run **or** --recon	Print command lines, but do not execute commands.
--no-print-directory	Turn off "-w" family of options, even if turned on implicitly.
-o *file* **or** --old-file=*file* **or** --assume-old=*file*	Assume that *file* is very, very old, and do not remake it.
-p **or** --print-data-base	Print GMAKE's internal predefined rules database which specify how to build target files from source files.
-q **or** --question	Do not build target, simply return exit code that specifies if target is up to date or not.
-r **or** --no-builtin-rules	Disable GMAKE's internal predefined rules which specify how to build target files from source files.
-s **or** --silent **or** --quiet	Do not echo commands as they are executed.
-S **or** --no-keep-going **or** --stop	Cancel "-k" family of options. Stop when target cannot be built.

Option		Description
-t --touch	**or**	Touch targets (update time/date stamp) instead of building them.
-v --version	**or**	Display GMAKE's internal version number
-w --print-directory	**or**	Print the current directory
-W file --what-if=*file* --new-file=*file* --assume-new=*file*	**or** **or** **or**	Consider file to always be new (always build target)
--warn-undefined-variables		Warn when undefined variables are referenced.

7.4.1 Additional Documentation

The GMAKE utility in the Merlin SDK is the GNU MAKE utility from the Free Software Foundation.

Additional documentation on the GNU MAKE utility, not specific to Merlin, is available separately.

7.5 Puffin Debugger

There are two flavors of the Puffin debugger for Merlin. One is a console-based application, the other is a MS Windows application. Neither version recognizes any arguments on the command line.

At the time this document was written, the graphic user interface version of the Puffin debugger was being rewritten. The goal is to enhance basic functionality and add features relating to C/C++ source level debugging. The new version will be almost entirely new in appearance and function, so in preparing this document we decided not to spend too much time trying to further document the older version.

7.5.1 Additional Documentation

The existing documentation on the PUFFIN debugger is available in the form of three separate documents:

- ***X-Lisp: An Object Oriented Lisp***
- ***X-Lisp Tutorial: An Introduction For C Programmers***.

- ***Puffin API***

Additional documentation regarding the updated version of the PUFFIN debugger will be released concurrently with the new version.

7.6 MLOAD Utility

The table below shows some of the more useful command line options for the MLOAD utility.

Option	Description
-?	Help. Display command line options.
-!	Reset the merlin chip
-~ *from : to : count*	Update host flash memory f(rom), t(o), c(ount) Note that this command affects' the host machine's flash memory, not the ACE360 board.
-a *file : address*	Load the specified *file* into host memory at the specified hexadecimal *address*.
-b *file : address*	Load the specified *file* into host memory at the specified hexadecimal *address* as a straight binary transfer. The file may contain anything.
-dm *address : num*	Dumps memory starting from the specified hexadecimal address.
-dr	Dump registers for the currently selected MPE
-f	Use the fast loader for the next file (default)
-gp *packet*	Send the specified GDB *packet*
-gr *request*	Send the specified GDB *request* and print response -grv = print the debug stub version number (if this command fails, your stub is really old and you should contact VM Labs immediately for a replacement)
-h	Stop the processor
-m	Monitor mpe exceptions after loading
-p *mpe*	Select the processor specified by *mpe* (0-3).
-r	Start processor running
-s	Use the slow loader for the next file
-u *file*	Transfer an ACE360 Flash memory update file. Note: This command is supported only by stub revisions dated April 6, 1998 or later.
-v	print the debug stub version number
-w	Wait one second

For example:

```
mload -p0 foo.mpo -r -p1 bar.mpo -r -m
```

This will load FOO.MPO into the Merlin's mpe 0 processor and start it running. It will also load BAR.MPO into the Merlin's mpe 1 processor and start it running, then it will monitor processor exceptions until both of those processors halt.

7.7 VMAR Utility

The VMAR utility allows you to create and modify library archive files. An archive is a single file which contains a group of smaller files, usually object modules containing compiled code and data which will be accessed by the linker.

A library archive created by VMAR is intended to make life easier for the programmer and the linker. The linker is designed to search an archive file, determine what pieces of code and data are contained in each individual module, and then extract only those modules which are required to successfully link a program.

In order to allow more efficient searching by the linker, the VMAR program can create an index of all the symbols contained in the individual object modules within the archive. Once created, this index is automatically updated when object modules are added or deleted.

Other types of files may also be stored in VMAR archives, but unless they are intended to be used by the linker this may not be the best choice, as no compression is done by VMAR.

The format of the VMAR command line is:

```
vmar [options][modifiers] ARCHIVE [member…]
```

The command *options* and *modifiers* are described in the tables below. The *archive* parameter indicates the filename of the archive to be used or created. The *member* parameter contains a single module name, or a list of module names.

Please note that all command line options and modifiers are case-sensitive, and only one command option may be specified per command line. Note that the leading "-" before a command option is accepted, but not required.

Option	Description
d	Delete the object module *member* from the archive. Multiple members may be specified. Examples: `vmar –d libmath.a cosine.o` `vmar –dv libmath.a cosine.o sine.o`
m	Moves the module(s) specified by *member* to the end of the archive. If certain symbols are defined in more than one module within a library, the order of those modules within the archive can make a difference in how programs are linked. Using the ‘a’, ‘b’, or ‘i’ modifiers with the “m” option will allow you to move the module to a specific location rather than the end of the archive. The example below would operate on the LIBMATH.A library, and move the module named fabs.o to the position following the sine.o module: Example: `vmar -ma sine.o libmath.a fabs.o`
p	Prints the specified modules to standard output. This would be used for text files such as source code which may be contained in an archive. This command does not produce readable output for a compiled object module. Example: `vmar p sine.c libmath.a`
q	Quick Append. Add the specified module to the end of the archive, without checking if there is another module of the same name already. The ‘a’, ‘b’, or ‘i’ modifiers do not affect this operation. Example: `vmar q libmath.a sine.o`

Option	Description
R	Insert the specified module into the archive, deleting any existing module with the same name. By default, modules are added to the end of the archive, but using the 'a', 'b', or 'i' modifiers will allow you to move the module to a specific location. Both examples below would operate on the LIBMATH.A library and add a module named sine.o. The first example would add it to the end of the library. The second example would add it to the library immediately before the arctan.o module: Example: `vmar r libmath.a sine.o` `vmar ri arctan.o libmath.a sine.o`
S	Create or update the library's object module index. Also available as a modifier which can be specified along with another command.
T	Display a table listing of the modules within the archive. The 'v' modifier will cause the permissions flag, timestamp, owner, group, and size information to be printed as well. Example: `vmar q libmath.a sine.o`
X	Extract the specified module from the library to an external file. If no module is specified, all modules within the library are extracted. If an external file with that name already exists, it is overwritten. The archive contents are not changed. Example: `vmar x libmath.a sine.o`

Modifier	Description
a *relpos*	Specify that the operation should happen at the position within the archive immediately after the specified *relpos* module. This modifier can be used with the "r" and "m' command options. Example: `vmar ra arctan.o libmath.a sine.o`

Modifier	Description
b *relpos* **or** i *relpos*	Specify that the operation should happen at the position within the archive immediately before the specified *relpos* module. This modifier can be used with the "r" and "m' command options. Example: `vmar rb arctan.o libmath.a sine.o` `vmar ri arctan.o libmath.a sine.o`
c	Disable warning when it's necessary to create the archive in order to make an update. Normally, if the archive does not exist, it is created when you attempt to add a module using the "r" command option, but a warning is issued. Using this modifier will disable the warning.
i	Specify that the operation should happen at the position within the archive immediately before the specified *RELPOS* module. This modifier can be used with the "r" and "m' command options. Example: `ar rb arctan.o libmath.a sine.o`
o	Preserve the original dates of modules when extracting them. Used with the "x" command option.
s	Create or update the library's object module index.
u	Conditionally add the specified files to the archive only if the timestamp is newer than the version of the file already in the archive.
v	Turns verbose mode on. More information gets printed about whatever command is being executed.
V	When used with any command option, causes the version of VMAR to be printed, instead of executing the command.

VMAR also has a mode which can be driven by either interactive commands or a script file. More details about this mode are listed below. The command line format for this mode is:

```
vmar –M [<script]
```

Note that the script file is specified using standard input redirection. During interactive use, VMAR prompts for input (the prompt is "AR >"), and continues executing even after errors. If you redirect standard input to a script file, no prompts are issued, and VMAR abandons execution (with a non-zero exit code) on any error.

The command language is not designed to be equivalent to the command-line options; in fact, it provides somewhat less control over archives. The purpose of the command language is to ease the transition for developers who already have scripts written for the MRI "librarian" program.

The command language syntax works according to the following rules:

- Commands are recognized in upper or lower case. For example, *LIST* is the same as *list*.
- Only one command per line, located at the beginning of the line.
- Empty lines are ignored.
- Comments begin with "*" or ";", everything afterwards on that line is ignored.
- A list of file names or module names may be separated by either spaces or commas.
- The "+" character is used to continue a command on the following line.

A basic reference to the available command is provided below. Note that most commands operate on the *current* library, which is the one last specified using either the OPEN or the CREATE command.

Command	Description
addlib *library*	Reads the archive specified by *library* and copies all modules into the current archive
addlib *module*	Adds the specified *modules* to the current archive.
Clear	Deletes all modules from the current archive.
Create *archivefile*	Creates a new archive with a filename of *archivefile*.
Delete *module*	Deletes the specified module from the current archive
Directory *archive* [(*modules)*] [*outputfile*]	Lists the modules contained in the specified *archive* file. If a module list is specified within parenthesis following the archive name, then only modules matching the list will be shown. The output may optionally be directed to a file by specifying the *outputfile* parameter.
End	Exit from VMAR with an exit code of zero. Does not automatically save changes to the current archive. If you do not use SAVE before END, then your changes are lost.
Extract *modulelist*	Extract the specified module(s) from the current archive and save them into the current disk directory.
List	Display the contents of the current archive.

Command	Description
Open *archive*	Opens a new archive.
Replace *module*	Replaces the specified module within the archive with a file of the same name in the current disk directory.
Save	Saves changes to the current archive. Any changes to the archive will not be permanent until this command is used.
Verbose	Turns on the verbose mode of each command.

7.8 VMNM Utility

The VMNM utility displays information about symbols contained in object modules or library archives.

The format of the VMNM command line is:

```
vmnm [options] [objfiles]
```

The *objfiles* parameter is a list of object modules or library archives created with VMAR. The command *options* are described in the table below. Please note that all commands and modifiers are case-sensitive:

Option	Description
-A **or** -o **or** --print-file-name	Print the module name next to each symbol, rather than just once at the top of each group.
-a **or** --debug-syms	Display special debugger-only symbols normally not listed.
-B	Specify BSD-format output. Same as "*–f bsd".*
-C **or** --demangle	Demangle encoded C++ names into user-level names.
-D **or** --dynamic	Display dynamic symbols
-f *format*	Specify output format. The *format* parameter should be "bsd", "sysv", or "posix". The default format is "bsd".
-g **or** --extern-only	Display only externally defined symbols
-n **or** --numeric-sort	Sort symbols according to address, rather than name
-p **or** --no-sort	Do not sort symbols. Output them in order found.

Option	Description
-P **or** -portability	Use Posix.2 format output. Same as "*-f posix*".
-s **or** --print-armap	When input file is a library, include the archive object module index.
-r **or** --reverse-sort	Reverse the order of the symbol sort.
--size-sort	Sort symbols by size. Size is determined by the next highest symbol.
-t *radix* **or** --radix=*radix*	Use *radix* for printing symbol values. Valid values are "d" for decimal, "o" for octal, or "x" for hexadecimal.
--target=*bfdname*	Specify a non-standard object file format
-u **or** --undefined-only	Display only undefined symbols (those external to object module or library).
-l **or** --line-numbers	For each symbol, use the debugging information to attempt to find the source filename and line number and print it. For undefined symbols, displays the source filename and symbol which references the symbol.
--version	Display the internal version number of VMNM.
--help	Display available command line options for VMNM.

7.9 VMOCOPY Utility

The VMOCOPY utility copies the contents of one object module to another, optionally performing a format conversion in the process.

The format of the VMOCOPY command line is:

```
vmocopy [options] infile [outfile]
```

The *infile* parameter specifies the source file. The *outfile* parameter is optional. If present, it defines the name of the output file. If absent, then a temporary file is created, and after processing is finished, the input file is overwritten.

The command *options* are described in the table below. Please note that all commands are case-sensitive:

Option	Description
--add-section= *sectionname= filename*	Add a new section named *sectionname* while copying the file. The contents of the new section are taken from *filename* and the size of the section will be the file size of that file. This is only supported by file formats which can support arbitrary section names.
--adjust-section-vma= *section{=,+,-}value*	Set or adjust the address of the specified *section*. If *section=value* is used, the address is set as specified. If *section+value* is used, the address is incremented by *value*. If *section-value* is used, the address is decremented by *value*.
--adjust-start=*increment*	Adjust the starting address of the file by adding *increment*. Note that some object file formats may not support this.
--adjust-vma=*increment*	Adjust the address of all sections, as well as the start address, by adding *increment*. Note that some object file formats may not support this.
--adjust-warnings	Issue warnings is a section specified via the "—adjust-section-vma" option does not exist. (default)
-b *byte* **or** --byte=*byte*	Discard file contents, except for every *byte*th byte. The *byte* parameter should be in the range of 0 to *interleavefactor*-1, where *interleavefactor* is the value specified using the "-i" command option, or the default value of 4. This option is used to create source files for multiple ROM or EPROM chips.
--debugging	Convert debugging information, if possible. Default for this option is FALSE.

Option	Description
-F fmt **or** --target=*fmt*	Specify that the original file uses the object code format specified by *fmt*, and rewrite it in the same format. If this option is not used, VMOCOPY will attempt to deduce the source format. See the ***File Format Types*** section below for more information.
-g **or** --strip-debug	Remove debugging symbols only
--gap-fill=*value*	Fill gaps between sections with the specified *value*. This is done by increasing the size of the section with the lower address, then filling in the gap with bytes containing *value*.
--help	Display help about command line options
-I *fmt* **or** --input-target=*fmt*	Specify that the input file uses the object code format specified by *fmt*. If this option is not used, VMOCOPY will attempt to deduce the source format. See the ***File Format Types*** section below for more information.
-i *interleavefactor* **or** --interleave=*interleavefactor*	Specify the interleave factor to use in conjunction with the "-b" command option..
-K *symbolname* **or** --keep-symbol=*symbolname*	Keep only *symbolname* from the source file. May be used multiple times to specify multiple symbols.
-N *symbolname* **or** --strip-symbol=*symbolname*	Remove *symbolname* from the source file. This option may be used multiple times for multiple symbols. May be combined with –K option.
--no-adjust-warnings	Disable warnings in the event that a section specified via the "—adjust-section-vma" option does not exist.
-O *fmt* **or** --output-target=*fmt*	Specify that the output file uses the object code format specified by *fmt*. If this option is not used, VMOCOPY will use the source format. See the ***File Format Types*** section below for more information.

Option	Description
--pad-to=*address*	Pad the output file until *address* is reached. The size of the last section is increased to match. The extra space is filled in with the value specified for the "—gap-fill" option, or the default of zero.
-R *sectname* **or** --remove-section=*sectname*	Remove the section specified by *secname* from the output file. This option may be used multiple times for different sections. Note that using this option incorrectly may make the output file unusable.
--remove-leading-char	Some object file formats use a special symbol at the start of every symbol, such as an underscore. This option will remove the first character from all global symbols.
-S **or** --strip-all	Remove all symbols.
--set-section-flags= *section=flags*	Set the flags for the specified *section*. The *flags* argument should be a comma-separated string of flag names: alloc, load, readonly, code, data, rom. Not all flags are supported by all object formats.
--set-start=*value*	Set the starting address of the file to *value*. Note that some object file formats may not support this.
-v **or** -verbose	Verbose output. List everything that is modified. For archives, this lists all members.
-V **or** --version	Display VMOCOPY's internal version number
-X **or** --discard-locals	Remove compiler-generated local symbols (usually starting with "L" or ".")
-x **or** --discard-all	Remove all non-global symbols.

7.9.1 File Format Types

Certain command options require that a file format be specified. This should be one of the formats shown in the table below:

Format Name	Description
oz-local-coff	Merlin-specific flavor of the COFF format (default file format)
srec	S Records

8. Merlin Development System Documentation

This chapter will provide an overview of other pieces of documentation that are available for the tools and libraries that make up the Merlin SDK.

Basic introductory documentation for the main tools is provided in chapter 0. For some tools, no further documentation is required. For others, the information in chapter 0 just scratches the surface.

Some of the documents mentioned in this chapter are available in hardcopy. Some are available in an online readable form, either as Adobe Acrobat files which can be viewed with the Adobe Acrobat reader, or else as HTML files which can be viewed with a web browser such as Microsoft Internet Explorer or Netscape Navigator. Contact VM Labs Developer Support if you are unable to view these files.

For each document, we'll list the title, the format(s) available, and give a brief description of the contents.

Documents in online format are always located in the VMLABS\DOC directory or a subdirectory, with the exception of the main SDK README file, which is located in the VMLABS directory.

The documents are grouped into separate sections for ***Tools***, ***System & Hardware***, and ***Libraries***. Documents regarding tools which are specific to a particular library are listed along with the library.

8.1 Tools

Chapter 0 includes a basic introduction and reference for several tools in the Merlin SDK. Additional documentation may also be found in the following documents.

Title	Description	Format
Optimizing Your LLAMA	A step-by-step tutorial to hand optimizing assembly code for Merlin using the Llama assembler	HTML
LLAMA User's Manual	This tells you everything there is to know about the Llama assembler.	Printed

Title	Description	Format
Merlin GCC README	Merlin-specific details about the C/C++ compiler	HTML
Using GNU CC	The basic non-Merlin specific user's manual for the C/C++ compiler.	Printed
GNU Make Manual	Details about the GNU Make utility.	Printed
Xlisp Manual	An introduction and manual for the X-Lisp programming language used by the Puffin debugger.	Printed
Xlisp Tutorial	A tutorial showing how to get the most out of the Xlisp programming language built into the Puffin debugger.	Printed
Puffin API	Basic documentation for the Xlisp API commands provided with the Puffin debugger.	HTML

8.2 System & Hardware

Title	Description	Format
BIOS Overview		HTML
Merlin Media Architecture Programmer's Guide	The basic documentation for the Merlin processor. Includes an assembly language reference, register documentation, and tons of other important details.	Printed
OZ Hardware Bug List	This list discusses the hardware bugs known to exist in the "Oz" revision of the Merlin processor. The "Oz" revision is the one used in the initial batch of developer kits and will be replaced by the "Aries" version in production hardware and later versions of the developer kit.	Printed

8.3 Libraries

Title	Description	Format
Merlin Troubleshooting	This document discusses a number of programming related problems and potential solutions	Printed
Jeff Minter's Object List API	Document describing the C/C++ level API for Jeff Minter's Object List sprite library	Acrobat
Merlin 2d Library	API for 2d graphics (lines, circles, text, etc.)	Printed
Merlin 3d Library	Details about the original Merlin 3d library.	Printed
Merlin Utility Functions Programmer's Manual	Details about the Merlin Utilities library	Printed
Merlin Synth API	Details about the Merlin Synth, a General MIDI compatible synthesizer.	Printed
Synth Voicing Tool	Information about using the synth voicing tool to create your own patches for the Merlin Synth.	Acrobat

This page intentionally left blank.

9. Running Your First Program

This section will walk you through the steps involved in compiling and executing your first Merlin sample program.

At this stage, you should have all of your hardware connected and configured as described in section 1. You must also have downloaded and installed the Merlin SDK as described in section 2.4.2

9.1 Check your configuration

If everything is installed and connected correctly, you should be able to reset the Merlin development system using the command line:

```
mload -!
```

The screen should blink, change to a color static-like display for a moment, then show a vertical color bar test pattern. If this fails, continue through the troubleshooting information below. Otherwise, you can skip ahead to section 9.2.

If your machine does not respond, cycle the power and try again. If it still doesn't work, then try this:

```
ping <ip address>
```

where <ip address> is the TCP/IP address that has been assigned to your Merlin development system. If the network connection is being located correctly, you'll get a message like this:

```
C:\vmlabs>ping 192.1.6.222

Pinging 192.1.6.222 with 32 bytes of data:

Reply from 192.1.6.222: bytes=32 time=6ms TTL=60
Reply from 192.1.6.222: bytes=32 time=2ms TTL=60
Reply from 192.1.6.222: bytes=32 time=5ms TTL=60
Reply from 192.1.6.222: bytes=32 time=4ms TTL=60
```

Note that the numbers returned may be different from machine to machine.

If you're able to "ping" the machine, then you should double check that the MD_PORT environment variable is set correctly as specified in section 6.1.

If your machine does not respond, you'll get a message like this:

```
C:\vmlabs>ping 192.1.6.226

Pinging 192.1.6.226 with 32 bytes of data:

Request timed out.
Request timed out.
Request timed out.
Request timed out.
```

This most likely indicates that the network configuration is not correct. In rare cases, it may indicate a hardware problem.

Make sure that everything is connected properly and configured correctly, as shown in section 0**.** If all else fails, contact VM Labs Developer Support as described in section 2.1.

9.2 Your First Sample

This section will walk you through the process of compiling and running a simple Merlin program written in C.

9.2.1 Compiling A Sample Program

1) Change to the \VMLABS\SAMPLE\C-SAMPLE directory. This folder contains a very simple "hello world" sort of sample program.

2) Execute the GMAKE utility. This utility will execute the C compiler to compile the source code according to the rules specified in the MAKEFILE. This should result in output that looks like this:

```
C:\vmlabs\sample\c-sample>gmake

mgcc -DSCRNWIDTH=360 -DSCRNHEIGHT=240
 -DDMA_XFER_TYPE=4 -O -g -c hello.c -o hello.o

mgcc -o hello.cof hello.o -lmutil
```

Please note that the line breaks will be formatted differently on screen, also an extra blank line has been added between commands for clarity.

Let's go over the output from step 2 and get into the details of what's being done.

The tool being called by GMAKE is the MGCC program, which is a driver program for the GCC C/C++ compiler. It provides a single interface to the C compiler, assembler, and linker. It's essentially equivalent to the basic "cc" program included with most command-line-oriented C/C++ compilers.

Here MGCC is being called twice in a row by GMAKE. Let's look at the first call and describe each of the options specified.

The command-line arguments starting with "-D" are equivalent to "#define" statements embedded at the top of the C source code. For example, the first one, "-DSCRNWIDTH=360" is equivalent to having "#define SCRNWIDTH 360" at the top of your source code file.

The "-O" option specifies that optimization should be used.

The "-g" option specifies that debugging information should be included in the final output file.

The part that reads "-c hello.c" specifies that the compiler should compile the source file hello.c and then exit, without calling the linker.

The last part of the line, "-o hello.o" specifies that the name of the output file being created should be HELLO.O.

Now let's look at the second call to the MGCC program again. This time, a different set of arguments is used.

The "-o hello.cof" argument specifies that the name of the output file should be HELLO.COF. This is followed by the name of the HELLO.O file, which will be passed to the linker to build an executable program file.

The "-lmutil" at the end of the line specifies that the linker should include the library file MUTIL.A in the link. This file is assumed to be located in the library directory specified by your DJGPP.ENV configuration file.

9.2.2 Running The Sample Program

1) Before you execute the sample, make sure the machine has been reset. You may cycle the power, or enter the command:

```
mload -!
```

This command will reset the machine and ensure that it is ready to download new code.

2) Now you can download the sample we just compiled by using the command:

```
mload -p0 hello.cof -r
```

The "-p0" portion of the command-line specifies that processor zero is the target for the options that follow. The "hello.cof" file is the name of the executable that was created in section 9.2.1. Finally, the "-r" option specifies that it should start running the code when the download is complete.

You'll note that the MLOAD program is used quite a bit. In fact, it is your main interface to the Merlin system other than the debugger.

9.3 Your Second Sample

Now we'll move over to another sample. This one isn't pretty, but it shows how C and assembly language can be combined.

9.3.1 Compiling Sample #2a

1) Change to the \VMLABS\SAMPLE\AUDIO directory.

2) Reset the machine.

3) Enter the command:

```
LOAD1
```

This is a batch file which will call the GMAKE tool to build the executable, and also download it immediately afterwards. This should result in output that looks like this:

```
C:\vmlabs\sample\audio>load1

C:\vmlabs\sample\audio>gmake -f game+eng.mak

mgcc -O   -c pcmtest.c -o pcmtest.o

llama -I../include -nologo -fcoff -c -o pcmcore.o pcmcore.s

llama -I../include -nologo -fcoff -c -o pcmlib.o pcmlib.s

llama -I../include -nologo -fcoff -c -o pcmdata.o pcmdata.s

mgcc -o pcmtest.cof pcmtest.o pcmcore.o pcmlib.o pcmdata.o
-lmutil
```

```
C:\vmlabs\sample\audio>mload -p0 pcmtest.cof -r
Loading 'pcmtest.cof' into mpe 0
Loaded in 4.611 seconds
Starting mpe 0
```

Again, line breaks have been added to make the output more readable here.

The call to GMAKE specifies that the "game+eng.mak" file should be used as the makefile by using the "-f" command line option.

Next, the C compiler is called to compile the PCMTEST.C source file.

On the next three lines, the LLAMA assembler is called to assemble the project's assembly language files. The "-I../include" argument specifies the location of the directory where the LLAMA assembler should search for included files that are not located in the current directory.

The "-nologo" argument does something which I'll have to look up.

The "-fcoff" argument specifies that the output file should be in COFF format.

The "-c" option specifies an option relating to cache usage which we'll ignore for the moment. (See the LLAMA documentation for more information.)

The "-o *file*" option on each line specifies the output filename to be used.

And each line ends with the name of the LLAMA source code file to be assembled.

Finally, the MGCC program is called again to link all of the object modules and associated library functions into an executable program named PCMTEST.COF.

9.3.2 Running Sample #2a

Running this sample is done automatically for you by the batch file, assuming no errors occur during the creation of the executable file.

This particular sample doesn't really do much in terms of a display, but it demonstrates how to play an audio sample using a simple PCM engine library.

This page intentionally left blank.

10. FAQ For New Developers

This chapter contains a variety of frequently asked questions on various topics, along with the appropriate answers.

Each question is only listed once, so if you don't find what you're looking for under one topic, make sure to check under related topics.

This chapter will be updated regularily. For more information relating to programming issues, please also see the separate document titled ***Merlin Troubleshooting Guide***.

10.1 Communicating With Merlin

Q: I have connected the Merlin development system to my network, but I cannot communicate with it from my host PC.

A: Most likely, there is a problem with the IP address setting of your Merlin development system. See chapter 1 for details.

Otherwise, it's possible that the MD_PORT environment variable is not set properly. See section 6.3 for details.

Q: I made a crossover network cable to connect my PC directly to the Merlin development kit, but they're not talking.

A: Making your own network cables, especially a crossover cable, is not recommended.

First of all, the connections for a crossover cable are not quite intuitive, since multiple connections must be switched. Secondly, a homemade cable is more likely to provide significant amounts of electrical noise that will cause network packet errors, slowing down transfers and communication.

10.2 Connections

Q: What's the miscellaneous video connector for? I don't seem to have a cable that will fit into it.!

A: This connector allows you to create your own cable for monitors with non-standard connections.

10.3 Tools

Q: Do the Merlin SDK tools work under Windows NT?

A: The tools are not officially supported, but the only known problem is that long filenames are not properly supported. The solution is to use 8.3 filenames under NT.

Q: Does the Merlin SDK include C++?

A: The libraries provided by VM Labs are C-oriented, but C++ is fully supported by the compiler for writing your own code.

Q: The C compiler is giving me a message saying "Virtual memory exhausted". What do I do?

A: The compiler likes lots and lots of memory. Your host computer should have at least 32mb of RAM, and 64mb won't hurt.

Secondly, if you're running an MSDOS Command prompt under Windows 95/98, then locate the command prompt icon, right-click it, and select "Properties." Under "memory", change the "DPMI" setting from "auto" to "65535". This will configure things so that the maximum possible amount of memory may be used by tools running under the command prompt.

10.4 Libraries

Q: Do the runtime libraries use multiple processors?

A: None of the libraries supplied by VM Labs use more than one processor without letting you know.

The MML3D library allows the programmer to specify how many processors will be used for the rendering process.

10.5 Video

Q: Can the Merlin development system be switched into PAL video mode?

A: Not currently. A future revision of the boot ROM will allow you to configure the video display to choose between NTSC and PAL video output.

Q: Can I use video input?

A: Video input is possible with the addition of a video encoder board to your Merlin development system. Contact VM Labs developer support for information about availability..

10.6 DVD Reading

Q: Are development units with DVD drives available yet?

A: Not as of this writing. DVD drives and the media access libraries are expected to be made available during Fall 1998.

In the meantime, take advantage of the fact that development systems have 16mb of EDO RAM instead of 4mb, and store information there that would otherwise be loaded from disc.

Alternately, you can use the PC File System library and utility to access files on the host PC's file system.

Q: Can I connect a DVD-ROM drive to my Merlin development kit?

A: No. The electronic interface for a DVD-ROM drive is designed to connect to a computer. Merlin expects a bare mechanism, and customized firmware is required for each different type.

10.7 Inter-Processor Communication

Q: Can a process running C code in one MPE start a process in another MPE?

A: Absolutely. This can be done using the ***StartMPE()*** function from the LIBMUTIL library.

Q: Can a process running C code in one MPE read or write data to the memory space of another MPE?

A: For regular internal memory, this can be done using the ***_mpedma()*** function. For the register space of another processor, use the ***_mpedmaregister()*** function. Both functions are from the LIBMUTIL library.

10.8 Memory Cards

Q: How much data can be stored on a memory card?

A: Memory cards will have a minimum of 2 megabytes of storage space.

11. Glossary

GCC — The Gnu C Compiler created by the GNU project of the Free Software Foundation. This compiler is available for most microprocessors and was adapted by VM Labs for the Merlin processor.

Llama — The Merlin assembler.

Merlin — This is the name of the VM Labs custom processor, and in a broader sense the name of the development system hardware.

MPE — An acronym for "multi-processing element". It refers to either one of the four separate modules of the *Merlin* processor, or to a software program intended to run in a single module.

MPO — Merlin Processor Object module. An object module output by the LLAMA assembler.

Oz — Code name for the current version of the Merlin processor

Puffin — This is the name of the debugger/emulator from the Merlin development system.

Puffinw — Version of Puffin designed to run under Microsoft Windows.

This page intentionally left blank.

12. Development System Files

The tables below will give a brief description of various files in the development system. It is expected that these files and subdirectories will be contained within the VMLABS directory on the root directory of a drive.

12.1.1 VMLABS\CFG Directory

VMLABS\CFG Directory	
File	**Description**
DJGPP.ENV	This file contains configuration settings used by the GNU GCC C/C++ compiler and other GNU tools.
PUFFINW.INI	This file contains the configuration settings used by the PUFFIN debugger.

12.1.2 VMLABS\BIN Directory

VMLABS\BIN Directory	
File	**Description**
ACELOAD.BAT	Batch file for updating Merlin ACE360 Stub
ACELOADR	File used for updating Merlin ACE360 Stub
AS.EXE	Assembler used by GCC compiler
CC1.EXE	GCC C compiler
CC1PLUS.EXE	GCC C++ compiler
COFF2MPO EXE	Converts a COFF executable to an MPO format executable
COFFDUMP.EXE	Dumps information about the contents of a COFF executable or object module file.
CPP.EXE	GCC C/C++ preprocessor
GMAKE.EXE	GCC Make utility
GMIDIRAM.SBI	Merlin synth wavetable samples
GOSYNTH.BAT	Batch file to load and start Merlin Synth.
LD.EXE	GCC linker
LLAMA.EXE	Merlin assembler.
MDEBUG.DLL	Puffin debugger DLL
MERLIN.DLL	Puffin debugger DLL
MG++.EXE	Part of GCC compiler.
MGCC.EXE	GCC compiler driver program (similar to GCC or CC)
MLOAD.EXE	Transfers binary code and/or data from PC to Merlin

VMLABS\BIN Directory	
File	**Description**
MMATH.DLL	Puffin debugger DLL
MPACKET.DLL	Merlin packet transfer DLL
MPO2COFF.EXE	Utility for converting MPO files to COFF format.
PUFFIN.EXE	MSDOS version of Windows 95 debugger
PUFFINW.EXE	Windows 95 version of Puffin debugger
QUICC.BIN	Merlin ACE360 Stub binary
REDIR.EXE	Utility for redirecting STDOUT and STDERR from compiler & other tools.
SENDMIDI.EXE	Merlin Synth MIDI playback tool
SENDMIDI.TCL	Tcl/TK file used by SENDMIDI program
SYNTHRAM.MPO	RAM-based version of Merlin Synth
SYNTHROM.MPO	ROM/FLASH ROM version of Merlin Synth
TRANSFER.MPO	Transfer routines which may be downloaded to merlin as needed
VMAR.EXE	Library archive utility (similar to AR)
VMLD.EXE	Linker utility (similar to LD)
VMNM.EXE	Library symbol lister utility (similar to NM)
VMOCOPY.EXE	
VMSREC.EXE	Utility to convert executable or object module to S-record format.
VMSTRIP.EXE	Utility to strip symbols & debugging information from executable program file
XLISP.DLL	XLISP interpreter DLL used by Puffin

12.1.3 VMLABS\INCLUDE Directory

Note: Some header files not required for Merlin have not been listed

VMLABS\INCLUDE Directory	
File	**Description**
_ansi.h	Standard C header file
_syslist.h	Standard C header file
ar.h	Standard C header file
assert.h	Standard C header file
ctype.h	Standard C header file
dirent.h	Standard C header file
errno.h	Standard C header file
fastmath.h	Standard C header file
fcntl.h	Standard C header file
files.txt	Standard C header file
float.h	Standard C header file

VMLABS\INCLUDE Directory	
File	**Description**
grp.h	Standard C header file
ieeefp.h	Standard C header file
iso646.h	Standard C header file
limits.h	Standard C header file
locale.h	Standard C header file
math.h	Standard C header file
merlin.i	Llama Assembler Include File
new.h	Standard C header file
paths.h	Standard C header file
process.h	Standard C header file
proto.h	Standard C header file
pwd.h	Standard C header file
reent.h	Standard C header file
regdef.h	Standard C header file
setjmp.h	Standard C header file
signal.h	Standard C header file
stdarg.h	Standard C header file
stddef.h	Standard C header file
stdio.h	Standard C header file
stdlib.h	Standard C header file
string.h	Standard C header file
syslimits.h	Standard C header file
termios.h	Standard C header file
time.h	Standard C header file
unctrl.h	Standard C header file
unistd.h	Standard C header file
utime.h	Standard C header file
utmp.h	Standard C header file
varargs.h	Standard C header file
va-merl.h	Standard C header file
va-merlin.h	Standard C header file

VMLABS\INCLUDE\MERLIN Directory	
File	**Description**
joystick.h	Standard C header file
m2prim.h	Standard C header file
m2pub.h	Standard C header file
m2types.h	Standard C header file
m3d.h	Standard C header file
m3dbuf.h	Standard C header file

VMLABS\INCLUDE\MERLIN Directory	
File	**Description**
m3dmat.h	Standard C header file
m3dtypes.h	Standard C header file
mcom.h	Standard C header file
merldma.h	Standard C header file
merlutil.h	Standard C header file
mlcolor.h	Standard C header file
mlexcept.h	Standard C header file
mlhost.h	Standard C header file
mlpixmap.h	Standard C header file
mltypes.h	Standard C header file
msprintf.h	Standard C header file
pcm.h	Standard C header file
sdram.h	Standard C header file

VMLABS\INCLUDE\JPEG Directory	
File	**Description**
CDERROR.H	Standard C header file
CDJPEG.H	Standard C header file
JCHUFF.H	Standard C header file
JCONFIG.H	Standard C header file
JDCT.H	Standard C header file
JDHUFF.H	Standard C header file
JERROR.H	Standard C header file
JINCLUDE.H	Standard C header file
JMEMSYS.H	Standard C header file
JMORECFG.H	Standard C header file
JPEGINT.H	Standard C header file
JPEGLIB.H	Standard C header file
JVERSION.H	Standard C header file

VMLABS\INCLUDE\MACHINE Directory	
File	**Description**
fastmath.h	Standard C header file
ieeefp.h	Standard C header file
setjmp-dj.h	Standard C header file
setjmp.h	Standard C header file
types.h	Standard C header file

VMLABS\INCLUDE\OBJC Directory	
File	**Description**
encoding.h	Standard C header file
hash.h	Standard C header file
NXConstStr.h	Standard C header file
objc-api.h	Standard C header file
objc-list.h	Standard C header file
objc.h	Standard C header file
Object.h	Standard C header file
Protocol.h	Standard C header file
sarray.h	Standard C header file
thr.h	Standard C header file
typedstream.h	Standard C header file

VMLABS\INCLUDE\SYS Directory	
File	**Description**
config.h	Standard C header file
dirent.h	Standard C header file
errno.h	Standard C header file
fcntl.h	Standard C header file
param.h	Standard C header file
reent.h	Standard C header file
resource.h	Standard C header file
signal.h	Standard C header file
stat-dj.h	Standard C header file
stat.h	Standard C header file
time.h	Standard C header file
times.h	Standard C header file
types.h	Standard C header file
unistd.h	Standard C header file
wait.h	Standard C header file
_types.h	Standard C header file

12.1.4 VMLABS\LIB Directory

VMLABS\LIB Directory	
File	**Description**
CRT0.O	Object module containing C runtime startup code. Linked in to execute before user code in program's *main()* function.

VMLABS\LIB Directory	
File	**Description**
CRTEND.O	Object module containing C runtime cleanup code. Linked in to execute after the end of the code in program's *main()* function.
LIBC.A	Standard C libraries.
LIBG.A	Standard C++ libraries
LIBGCC.A	GCC runtime functions
LIBJPEG.A	JPEG image decompression library (source code contained in SRC\JPEG subdirectory)
LIBM.A	Floating point math function library
LIBMML2D.A	2D graphics library (source code contained in SRC\MML2D subdirectory)
LIBMML3D.A	3D graphics library (source code contained in SRC\MML2D subdirectory)
LIBMUTIL.A	Utility function library (source code contained in SRC\UTIL subdirectory)

13. Hardware Bug List

Last update: June 1, 1998

This list contains all the known bugs in the Oz (Merlin Beta) silicon as of the above date. For each bug, there is a "level" code which indicates the type and seriousness, as shown in the table below:

Bug Level	Description
0	A quirk that probably ought to be fixed.
1	This bug can be worked around in software without significant system overhead.
2	This bug can be worked round, but has a significant system impact.
3	This bug cannot be entirely worked around, and could mask further bugs.

Also shown is the status of the bug for the next design revision of the Merlin chipset, known as *Aries***.**

Each bug has a basic description of the symptoms. A work-around for each problem is given where available.

Keep in mind that many of the hardware bugs are important only if you are designing a system around the Merlin chipset, and do not affect application programming.

13.1 MPE

The "ls" condition code is encoded wrong	
Level	1
Aries status	
Description	The "ls" condition code is implemented as C.Z, but it should be C+Z to correctly perform this logical function.
Workaround	Do not use the "ls" condition code.

Store pixel only works in YCrCb mode	
Level	1
Aries status	

Store pixel only works in YCrCb mode	
Description	If you attempt to store a pixel in GRB mode (*chnorm* clear) an incorrect value is written. If *chnorm* is set then store pixel behaves correctly.
Workaround	Always work with *chnorm* set (YCrCb mode).

MPE I-cache Repeated-Store	
Level	1
Aries status	
Description	This hardware bug can cause repeated execution of a "store" or "load" operation if it "aligns" in a certain way with a stall caused by an I-cache miss. In most cases, there is no harm done by this repeated execution. However, for certain registers, a repeated store or load operation can cause incorrect results and likely crash the program. The main danger is to I-cached code which stores directly to the *mdmacptr*, *odmacptr*, or *commxmit3* registers, or which loads directly from the *commrecv3* register. (Note there are some contrived situations in which problems could occur for I-cached code which stores to *mpectl*, *excepsrc*, *excepclr*, *excephalten*, and *icachectl* -- although these unlikely operations are already awash in other dangers.)

MPE I-cache Repeated-Store	
Workaround	This bug does not affect store operations in code running from local (non-I-cacheable) addresses. So in situations where part of the MPE's IRAM has been dedicated to non-cached code, the bug can be avoided by doing stores to the "dangerous" registers only from non-icached code. In fact, this is a natural arrangement for many applications, since it often makes sense to keep ISR's and DMA handlers local rather than cached. (A similar workaround is to execute the dangerous stores from local DTRAM rather than local) This bug will be made at least partially transparent to programmers by an assembler change that is planned. By extending the automatic I-cache padding rules the assembler already follows, stores and loads to "dangerous" registers can be positioned such that a bad alignment with an I-cache miss is not possible. Some situations, such as stores in branch delay slots and indirect stores, may have to be handled with assembler directives or other manual fixes.

MPE I-cache Overlay Stalls	
Level	1
Aries status	
Description	This hardware bug causes icached program execution to stall incorrectly during dma access to the IRAM. This does not affect most normal uses of the MPE I-cache. I believe it causes only a few esoteric restrictions: it means that an icached program should not perform an explicit dma to bring a code overlay into its own IRAM (a strange concept actually); it means that while an MPE is running an icached program, other processors should not dma into any part of that MPE's IRAM; and finally it means that icached programs should not put dma commands in IRAM (which, if you noticed the MPE's Harvard-like architecture, you probably thought was impossible anyway).

MPE I-cache Overlay Stalls	
Workaround	The only workaround is to avoid dma access to the IRAM when an icached program is running. For the specific problem of how to move an MPE from icached execution to local execution when all the IRAM is dedicated to the I-cache, there are a number of solutions. Probably the most elegant is to dma a small program to DTRAM, then jump directly there to execute it; that program can then dma some other code to the base of IRAM and jump directly there, thus converting to local non-I-cache execution.

13.2 Main Bus DMA and SDRAM Interface

The priority gateways in the bus arbiter must always be open	
Level	1
Aries status	
Description	The gateways which allow lower bus-request priority levels to have slots at higher round-robin priority levels must always be open; because if they are closed the lower priority level is never granted the bus.
Workaround	Leave the gateways open.

Audio DMA corrupts relative DMA on MPEs 1, 2, & 3	
Level	1
Aries status	fixed
Description	This bug causes relative MPE addresses to be converted to absolute addresses if an audio command fetch follows an MPE command fetch. This means they do not work properly (the relative address looks like an MPE 0 address if treated as an absolute address).
Workaround	All transfers by MPEs 1-3 must have the REMOTE bit set, with the internal address field modified accordingly, if audio DMA is enabled. This means that if audio DMA is running, then MPE 3 may not cache from main bus DRAM.

Audio DMA can hang the DMA controller	
Level	2
Aries status	fixed
Description	Under certain circumstances audio DMA can completely hang the DMA controller.

Audio DMA can hang the DMA controller	
Workaround	Transfer audio using the comm. bus. A more complex software Workaround is possible using software to restart the DMA. Consult VM Labs for further details.

Reading from MPE (external) to MPE (internal) does not work	
Level	1
Aries status	fixed
Description	Reading from MPE (external) to MPE (internal) does not work. This only affects mpe to mpe reads, not write DMA.
Workaround	You can reverse the direction of the transfer so that it is always write DMA. Where you would have performed a read, make the following changes to the command: 1) Swap the addresses (internal/external) in the DMA command, 2) Set the remote bit, and make sure that the internal address is relative to the 32 bit main bus address space 3) clear read (flags[13]).

A 4 MB SDRAM configuration using 2 x8 DRAMs only has 2 MB available	
Level	1 - hardware only
Aries status	Fixed
Description	When a system is configured to two 2M x 8 DRAMs to give 4 MB of DRAM, only the first 2 MB works. The upper and lower DQM (data mask) signals only go active for the low 2MB.
Workaround	The DQM signals for the upper 2MB are actually available on the high address lines, and could be combined with the others using fats AND gates if required.

A 4 MB SDRAM configuration using 2 x16 DRAMs only has no refresh in the high 2MB	
Level	1 - hardware only
Aries status	fixed
Description	When a system is configured to two 2M x 16 DRAMs to give 4 MB of DRAM, refresh only occurs for the first 2 MB.

A 4 MB SDRAM configuration using 2 x16 DRAMs only has no refresh in the high 2MB	
Workaround	If more than 2MB is needed software will have to perform refreshes.

16-bit filtered pixels show rounding errors	
Level	1
Aries status	fixed
Description	When applying vertical filtering to 16-bit pixels, artifacts can appear because of the limited precision available, most noticeably a green shift on white,
Workaround	Avoid vertical filtering on 16-bit pixels.

DMA Reads outside of DRAM can hang MPE software	
Level	1
Aries status	?
Description	If a DMA read outside the valid SDRAM area is performed, the DMA does not decrement the active count, although it does flag an exception. This may hang the MPE software.
Workaround	Restrict DMA to within the valid DRAM (wow!)

MPEG Swath Mode Vertical Filter cannot cross swath boundaries	
Level	0
Aries status	unchanged
Description	When applying a vertical filter to MPEG data in swath mode, if the pixels required would cross a swath boundary then the DMA will clip to pixels within the current swath.
Workaround	Restrict vertical scaling in this mode so that the pixel sampling points all lie within a swath. This means, in a pure swath based architecture, the vertical decimator is restricted to 8:7, 8:6 (or 4:3), 8:5 and so on till the limits of bandwidth.

MPE Instruction Tags cannot be accessed using Main Bus DMA	
Level	0
Aries status	not fixed
Description	Main bus DMA to the instruction tags causes a DMA exception
Workaround	If you need DMA access, use the Other Bus

13.3 Other Bus

Read data can be lost if other MPE DMA transfers are active	
Level	2
Aries status	fixed
Description	If an other bus read transfer happens at the same time as main bus DMA (or BDU DMA in MPE 1), then it is possible under certain circumstances for a long-word of read data to be lost from the other bus transfer. The remainder of the transfer will be shifted down one long-word. This does not appear to affect other bus transfers for system bus DRAM (in internal mode only) that do not cross DRAM< page boundaries.
Workaround	Avoid performing other bus read transfers during other DMA activity.

Transfers to another MPE that is using the Other Bus can hang	
Level	1
Aries status	fixed
Description	If you want to transfer data to or from another MPE you may only use the other bus if that other MPE is not using its other bus interface at all. It is possible for the other bus to hang if you break this rule.
Workaround	Use the main bus for MPE to MPE transfers. But note the limitation above under “Reading from MPE (external) to MPE (internal) does not work”.

Spurious Done Interrupts can be generated	
Level	1
Aries status	Fixed
Description	The done interrupt is generated not only when a requested DMA transfer completes but also when another MPE or the debug unit transfers data to or from your MPE.
Workaround	Poll for DMA completion.

13.4 Communication Bus

Receive buffer full flag is set too soon	
Level	0
Aries status	unchanged

Receive buffer full flag is set too soon	
Description	The receive buffer full flag is set as soon as the first long word of data is received. This means that the complete receive buffer register is not valid for a further three clock cycles.
Workaround	Care should be taken when optimizing code to ensure that you don't read the data until at least three clock cycles after you have ascertained that there is data to be read. Interrupt code is not affected as you can't get into an interrupt service routine that fast.

13.5 VDG

ovlCtrl, *pixHscale* and *vidCtrl* registers are effectively write only	
Level	1
Aries status	fixed
Description	The *ovlCtrl*, *pixHscale* and *vidCtrl* registers cannot be read correctly.
Workaround	Don't read them!

The horizontal 4-tap filter is always on	
Level	0
Aries status	fixed
Description	There is no mechanism for by-passing the horizontal filter, so a low-pass filter is always applied to main channel pixels.
Workaround	None required or possible.

The horizontal sync position is fixed	
Level	1 - hardware only
Aries status	fixed
Description	*Syn_Hstart* does not work correctly. *HSYNC* changes polarity when *Hcount* refreshes. Note that *Syn_Hend* is working correctly though.
Workaround	This probably restricts us to using a digital video encoder which uses *SAV* and *EAV* for line timing and synthesizes *HSYNC*.

Communication bus reads can be corrupted	
Level	1
Aries status	fixed

Communication bus reads can be corrupted	
Description	If the VDG comm. bus interface is waiting to transmit a response to a read command, it will accept another comm. bus command, which can corrupt the pending read response. The correct behavior is for it to keep its receive buffer flagged as full until the response has been successfully transmitted.
Workaround	The VDG comm. bus interface must be controlled from a single MPE. When it is waiting for a read response it must not send any further comm. bus commands.

Horizontal scaling is limited at 3:1 (image shrinking)	
Level	2
Aries status	fixed
Description	An image may not be horizontally shrunk by more than 3:1 in the output channel.
Workaround	Limit scaling to this, or scale the image using other means.

The main video channel cannot be turned off beneath an overlay	
Level	1
Aries status	fixed
Description	Once the main channel is activated, it will obtain the highest priority in accessing the Video FIFO. However, even if the channel is shut off after a certain number of lines, it will still request words from the FIFO. This means that the main channel will block off all requests from the overlay channel. Hence, if the screen is split vertically between these channels, it will not work, unless the overlay channel precedes the main channel.
Workaround	Do a narrow main channel under the overlay channel, it will not produce any visual distortion, but you will take a minor hit on the bandwidth and coding.

The pll_ref signal is disabled by reset	
Level	1 - hardware
Aries status	fixed
Description	The pll_ref output is disabled during reset. This means that external VCO clock sources will lose lock during reset.
Workaround	

Video pipes are initialized wrong	
Level	1
Aries status	fixed
Description	The pipes for each video channel (main, overlay and sub-picture) are initialized at the beginning of each video line regardless of the active state of that channel. This means each channel must be set to be the same number of lines as the active screen area,
Workaround	

Horizontal scaling does not work for all scaling values	
Level	2
Aries status	fixed
Description	Horizontal scaling has a pipelining problem which causes vertical stripes to appear on the screen at some scaling values.
Workaround	These scaling ratios are known to work: in : out 1 : 1 direct 3 : 4 pan/scan for MPEG 1 : 2 games 3 : 1 shrunk picture

Sub-Picture Control and Data FIFO settings are locked together	
Level	3
Aries status	fixed
Description	The sub-picture control and data channel FIFO settings can only be set to the same values, so these channels are not properly independent.
Workaround	None

Sub-Picture Run-to-the-end-of-line Code does not work	
Level	2
Aries status	fixed
Description	The sub-picture data run-length code for run to the end of line does not produce the correct colors.
Workaround	Replace the run-to-end-of-line code with regular run-length codes.

Sub-Picture Data DMA can not be stopped at the end of a field

Sub-Picture Data DMA can not be stopped at the end of a field	
Level	2
Aries status	fixed
Description	The software controlling sub-picture is not aware of the length of the sub-picture data for one field, and so it programs the length of the largest possible field. At the end of the field, it cannot stop the DMA to prevent the rest of the data being transferred.
Workaround	Software must parse the data length and program the hardware accordingly.

The FIFOs for each channel cannot be independently flushed	
Level	0
Aries status	fixed
Description	There is no mechanism to independently flush the FIFO channels.
Workaround	None

Vertical Filtering sometimes gives incorrect results	
Level	1
Aries status	fixed
Description	Some forms of vertical filtering produce visual artifacts, such as black spots in areas that should be white. This occurs because the horizontal filter can overflow when presented with input pixels of $FF, which the vertical filter generates.
Workaround	Avoid these vertical filter modes.

13.6 VIDEO IN

Video In has a variety of problems	
Level	1
Aries status	completely redesigned
Description	The Video In channel has a variety of problems and should only be operated with the driver developed by VM Labs. The individual issues are not discussed here.
Workaround	Use VM Labs' driver code.

The video in channel in pass-through mode captures data with the wrong clock	
Level	1 - hardware only

The video in channel in pass-through mode captures data with the wrong clock	
Aries status	mode no longer supported
Description	Video bypass mode has asynchronous logic which induces *VICLK* and *CLK* phase dependence.
Workaround	Phase relationship of *VICLK* must be controlled relative the main clock (visible as *SYSBCLK*).

13.7 GENERAL I/O

Timer 1 is unusable	
Level	1
Aries status	fixed
Description	It is not possible to write to timer 1, because both the timer 0 and timer 1 addresses actually perform a write to timer 0.
Workaround	Do not use timer 1.

Controller (joystick) receive data full flag cannot be cleared by polling	
Level	1
Aries status	fixed
Description	If receive data is read from the controllers by polling the interface the receive buffer full flag is not cleared. The only way to clear it is to receive data from the controller interface automatically by setting the commSend bit in the *ctrlStat1* or *ctrlStat2* registers.
Workaround	Do not poll to receive data, let the controller interface send it automatically over the comm. bus.

13.8 ROM Interface

ROM data bus can have contention	
Level	1 - hardware only
Aries status	fixed
Description	If the ROM idle pause time in the ROM control register is set to non zero, then the ROM control lines can go into an illegal state during this period which will cause contention between Oz and the ROM.

ROM data bus can have contention	
Workaround	The “tester mode” flag must always be set, and a ROM should be used whose tri-state disable time is of the order of 18 ns. This may be a problem, in which case we may have to use fast control logic on the OE pins of the bus switch to disable the ROM. As the chip boots with this function enabled we must place padding resistors on the ROM data output to limit contention currents, probably about 330 ohms.

Flash memory cannot be programmed	
Level	1
Aries status	fixed
Description	Flash memory programming requires single byte read and write transfers. These are not possible as the other bus only has long word granularity.
Workaround	External hardware would be required.

13.9 System Bus Interface

External mode DRAM bank control bits are split between host space and Merlin space	
Level	0
Aries status	?
Description	The bank control bits, which are used to govern the multiplexing performed when the SAMUX function is used, are split between the host space registers and the Merlin internal system bus control registers. Also, the mask bits on the bank1 address field are superfluous.
Workaround	Merlin and the host will have to co-operate on setting up this function for External Mode systems.

Mechanism for driving SYSBB high is marginal	
Level	1 - hardware only
Aries status	fixed
Description	SYSBB is momentarily driven high at the end of being driven low by Oz. The mechanism used (to avoid negative clock edge logic) does not allow it to achieve a proper high level before it stops driving.
Workaround	

DRAM Control Signals do not have enough drive	
Level	0 - hardware only
Aries status	fixed
Description	The internal mode DRAM control signals should have an increased drive strength.
Workaround	Buffer them externally if required.

Internal Mode DRAM Page Break mechanism can cause transfers to fail	
Level	1
Aries status	fixed
Description	There is a problem that can occur during an Other bus DMA cycle to system bus DRAM (in internal mode only) that can cause the next DMA to fail. The problem occurs when a read or write DMA has occurred with the last address being one less than that of the last location in a page. This will set a condition that anticipates a page break but is never cleared because the page boundary is not encountered. This erroneous condition will remain set until the next DMA is processed. This second one may have a premature termination to its cycle and cause errors.
Workaround	Avoid ending system bus DRAM transfers one long word before the end of the page. For example, cache accesses will never encounter this problem. If this cannot be avoided, then have a single access read DMA clear the erroneous condition. This DMA will have to have an address that will not induce the bug such as the start of the current page. DMA's that can produce this bug need to be identified and have the singular read done after the "offending" DMA was done. The singular read will read some data and clear the bug so that the next DMA that is requested can complete properly.

13.10 Audio Interface

SPDIF Channel Status Word is put out across both sub-frames	
Level	1
Aries status	fixed

Description	The SPDIF channel status word is output at the rate of one bit per sub-frame. In fact, the channel status is a separate stream for each of the two channels, and so should be output at the rate of one bit per sub-frame per channel.
Workaround	Only the first sixteen bits of channel status can be generated (all the rest are zeroes), and bit pairs should be duplicated in the channel status register to control this.

Audio In is turned on at reset	
Level	0
Aries status	fixed
Description	As it comes out of reset, the audio in channel will start sending packets to MPE 0 if it has a clock on its bit clock pin.
Workaround	It can be disabled by writing 63 to the *dataDelay* register. Boot ROMs created after 28 Jan 98 contain this code.

Audio In data is one-bit misaligned	
Level	1
Aries status	fixed
Description	Audio input data is captured one bit later than the *dataDelay* value set.
Workaround	Generally, -1 can effectively be set as the *dataDelay* when required as long as the width of the word strobe pulse is known. The data is then latched correctly by setting *dataDelay* one less than it should be.

13.11 Reset

The Configuration bits are reloaded on an external host reset	
Level	1 – hardware only
Aries status	fixed
Description	The power on configuration bits are reloaded on an external host reset. This is dangerous as the reset length is short, and the lines may not settle in time. The configuration should only be loaded on a hardware reset.
Workaround	Make sure that the configuration bits on *X_vdata* can settle in about 100ns.

This page intentionally left blank.

14. Software Bug List

This section has just gotten started. The bug lists are not yet comprehensive.

Eventually, this section will contain a comprehensive list of bugs known to exist in the SDK tools and libraries.

Last update: August 12, 1998

14.1 C/C++ Compiler

The 'long long' data type is not handled correctly.	
Description	Using the 'long' modifier on the 'long' data type to get a 64-bit integer data type is not handled correctly.
Workaround	Don't use "long long"

#include files not found when running under Windows NT	
Description	Error messages complain that files specified with the #include directive cannot be found, but those files are not missing, and the source file compiles without errors under Windows 95. The problem is that long filenames are not supported properly under Windows 95. If a filename is longer than 8.3 characters (8 character filename + 3 character extension), the file will not be located correctly. This also applies to the individual directory names specified in the search path,
Workaround	Make sure all directory and file names used by the compiler are no longer than the 8.3 recognized by standard MSDOS.

The '-minscrutable' option does not work right.	
Description	The '-minscrutable' option is supposed to cause the compiler to avoid using a frame pointer. However, the resulting code does not work properly.
Workaround	Don't use this option.

Code in an asm() block may fail	
Description	Code contained within an asm() block does not get protected from structural hazards in the surrounding code.
Workaround	For now, start any asm() with a couple of NOPs.

14.2 Llama Assembler

Llama generates spurious "too many instructions in packet" errors.	
Description	Llama does not resolve labels until its final pass; as a result, it must initially assume that operands involving labels may require 32 bits. This can lead to spurious errors about too many instructions in a packet using the 32 bit instruction prefix.
Workaround	None at this time.

Llama may reschedule instructions incorrectly	
Description	LLAMA assumes that external subroutines do not end with two tick operations. If this is not true of a subroutine, then under certain circumstances LLAMA might re-schedule instructions to use the result of a subroutine before it is ready. Obviously this bug is only a potential hazard if optimization is enabled, and only if LLAMA hasn't seen the actual subroutine definition itself and hence can't determine whether the subroutine contains a two tick operation.
Workaround	If the problem occurs, define the subroutine earlier in the source code, prior to the subroutine call.

Labels may not be specified within packets	
Description	Labels may not appear inside of packets. This is theoretically possible hardware-wise, but a good method to allow Llama to do this has not been found.
Workaround	None at this time.

14.3 Linker

This section under construction.

14.4 Puffin Debugger

This section under construction.

This page intentionally left blank.