```
          code for executing a profile and tracing
         captured in this example could be used for tuning th
       trace buffer wrap mode to a one time trace
     raceMode (hAxis1, PMDTraceOneTime);
    t the processor variables that we want to capture
  tTraceVariable (hAxis1, PMDTraceVariable1, PMDAx
 etTraceVariable (hAxis1, PMDTraceVariable2, PMDAxi
SetTraceVariable (hAxis1, PMDTraceVariable3, PMDAxis
// set the trace to begin when we issue the next update command
SetTraceStart (hAxis1, PMDTraceConditionNextUpdate);
// set the trace to stop when the MotionComplete event occurs
SetTraceStop (hAxis1, PMDTraceConditionEventStatus,
  PMDEventMotionCompleteBit, PMDTraceStateHigh);
etProfileMode (hAxis1, PMDTrapezoidalProfile);
  et the profile parameters
   Position(hAxis1, 200000);
    elocity(hAxis1, 0x200000);
      eleration(hAxis1, 0x1000);
        eration(hAxis1, 0x1000);
```

C-Motion Engine Development Tools

Manual

Revision 1.3/ April 2024

Performance Motion Devices, Inc.

80 Central Street, Boxborough, MA 01719

www.pmdcorp.com

NOTICE

This document contains proprietary and confidential information of Performance Motion Devices, Inc., and is protected by federal copyright law. The contents of this document may not be disclosed to third parties, translated, copied, or duplicated in any form, in whole or in part, without the express written permission of PMD.

The information contained in this document is subject to change without notice. No part of this document may be reproduced or transmitted in any form, by any means, electronic or mechanical, for any purpose, without the express written permission of PMD.

Warranty

Performance Motion Devices, Inc. warrants that its products shall substantially comply with the specifications applicable at the time of sale, provided that this warranty does not extend to any use of any Performance Motion Devices, Inc. product in an Unauthorized Application (as defined below). Except as specifically provided in this paragraph, each Performance Motion Devices, Inc. product is provided "as is" and without warranty of any type, including without limitation implied warranties of merchantability and fitness for any particular purpose.

Performance Motion Devices, Inc. reserves the right to modify its products, and to discontinue any product or service, without notice and advises customers to obtain the latest version of relevant information (including without limitation product specifications) before placing orders to verify the performance capabilities of the products being purchased. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgment, including those pertaining to warranty, patent infringement and limitation of liability.

Unauthorized Applications

Performance Motion Devices, Inc. products are not designed, approved or warranted for use in any application where failure of the Performance Motion Devices, Inc. product could result in death, personal injury or significant property or environmental damage (each, an "Unauthorized Application"). By way of example and not limitation, a life support system, an aircraft control system and a motor vehicle control system would all be considered "Unauthorized Applications" and use of a Performance Motion Devices, Inc. product in such a system would not be warranted or approved by Performance Motion Devices, Inc.

By using any Performance Motion Devices, Inc. product in connection with an Unauthorized Application, the customer agrees to defend, indemnify and hold harmless Performance Motion Devices, Inc., its officers, directors, employees and agents, from and against any and all claims, losses, liabilities, damages, costs and expenses, including without limitation reasonable attorneys' fees, (collectively, "Damages") arising out of or relating to such use, including without limitation any Damages arising out of the failure of the Performance Motion Devices, Inc. product to conform to specifications.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent procedural hazards.

Disclaimer

Performance Motion Devices, Inc. assumes no liability for applications assistance or customer product design. Performance Motion Devices, Inc. does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of Performance Motion Devices, Inc. covering or relating to any combination, machine, or process in which such products or services might be or are used. Performance Motion Devices, Inc.'s publication of information regarding any third party's products or services does not constitute Performance Motion Devices, Inc.'s approval, warranty or endorsement thereof.

Patents

Performance Motion Devices, Inc. may have patents or pending patent applications, trademarks, copyrights, or other intellectual property rights that relate to the presented subject matter. The furnishing of documents and other materials and information does not provide any license, express or implied, by estoppel or otherwise, to any such patents, trademarks, copyrights, or other intellectual property rights.

Patents and/or pending patent applications of Performance Motion Devices, Inc. are listed at https://www.pmdcorp.com/company/patents.

Related Documents

Magellan Motion Control IC User Guide

Complete description of the Magellan Motion Control IC features and functions with detailed theory of its operation.

C-Motion Magellan Programming Reference

Describes C-Motion Magellan Motion Control IC commands, with coding syntax and examples, listed alphabetically for quick reference.

C-Motion/PRP Programming Reference

Describes C-Motion language function calls and associated PRP-formatted packets along with data types for communication between the host and ION/CME or Prodigy/CME boards (other than ION/CME N-Series ION Digital Drives).

C-Motion/PRP II Programming Reference

Describes C-Motion language function calls and associated PRP-formatted packets along with data types for communication between the host and ION/CME N-Series ION Digital Drives.

ION/CME N-Series Digital Drive User Manual

How to install and configure ION/CME N-Series Digital Drives.

ION/CME Digital Drive User Manual

How to install and configure ION/CME 500 Digital Drives.

Prodigy/CME PC/104 Motion Board User Guide

How to install and configure Prodigy/CME PC/104 Motion Boards.

Prodigy/CME Stand-Alone Motion Board User Guide

How to install and configure Prodigy/CME Stand-Alone Motion Boards.

Prodigy/CME Machine-Controller Motion Board User Guide

How to install and configure Prodigy/CME Machine-Controller Motion Boards.

Table of Contents

This page intentionally left blank.

1. Introduction

In This Chapter

1.1 Introduction

This manual documents the C-Motion Engine Development Tools (also referred to as CME Development Tools). These tools are provided with CME devices and allow user application code to be created and compiled on a host PC, and then downloaded, executed, and monitored on the CME device's C-Motion Engine module.

The C-Motion Engine Development Tools provides the following features:

- Complete toolset for creation of user-specific applications running on the C-Motion Engine
- Open source compiler & motion control C libraries
- Interactive Development Environment
- Supports SPI (Serial Peripheral Interface), RS232/RS485, CANbus, Ethernet, and PC/104-bus communications

1.2 C-Motion Overview

The system used to program PMD products is called C-Motion. C-Motion provides a convenient set of callable routines comprising the C language code required for controlling PMD controller products, whether running on a separate host computer such as a PC, an embedded microcontroller, or on a C-Motion Engine. C-Motion includes the following features:

- Axis virtualization
- Ability to communicate to multiple PMD motion boards or modules
- Ability to communicate via PC/104 bus, serial, CAN, Ethernet, SPI (Serial Peripheral Interface), or 8/16 bit parallel bus
- Provided as source code, allowing easy compilation & porting onto various run-time environments including a PC, microprocessor, embedded card, or C-Motion Engine
- Can be easily linked to any C/C++ application

1.2.1 C-Motion Application Code

Broadly speaking there are two different ways user application code written in C-Motion can be used to control a PMD controller; the user application program can run on a host separate from the PMD controller, or the user application program can run directly on the PMD controller in a code execution module called the C-Motion Engine.

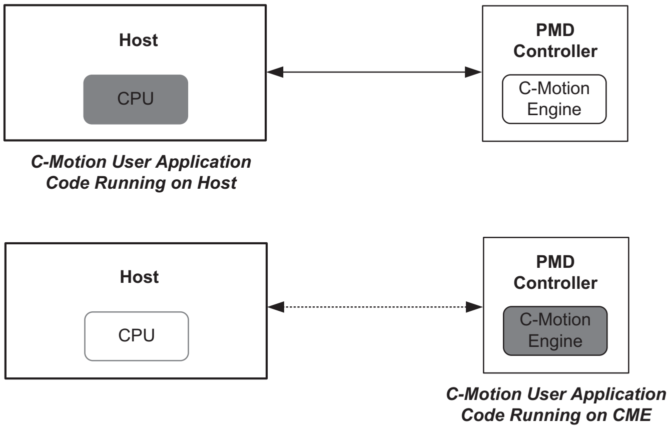
If the user code runs on a host separate from the PMD controller there are several types of possible hosts and several types of possible PMD controllers. If the PMD controller is a motion IC such as the Magellan Motion Control IC running on a user-designed motion board the host is often a microcontroller located on the same board that the motion IC is located on. In this scenario communications between the host microcontroller and the Magellan IC occur via SPI (Serial Peripheral Interface), CAN, serial, or parallel word.

The host controller may also be external to the PMD-based controller. In this case the host controller is generally a PC, a PLC, or a custom control board which communicates to the PMD-based controller via CAN, serial, or Ethernet, or via a board bus format such as PC-104. In this configuration the PMD-based controller may be an off-the-shelf product such as PMD's ION Digital Drive or Prodigy Motion Control board, or it may be a user-designed board with Magellan Motion Control ICs installed on the board.

If the user code runs directly on the PMD controller than this always occurs via the C-Motion Engine module. PMD products which support a C-Motion Engine have a "CME" in their product name, for example ION/CME N-Series ION Digital Drive.

In the next few sections we will talk more about how PMD controllers support these two different user application code modes (host-based or C-Motion Engine-based) and illustrate a powerful feature of C-Motion which is that once written, this same code, without modification, can be run on a host or in a C-Motion Engine.

Figure 1-1: Host-based versus C-Motion Engine Embedded User Application Code Execution


Host
CPU
PMD Controller
C-Motion Engine
C-Motion User Application Code Running on Host
Host
CPU
PMD Controller
C-Motion Engine
C-Motion User Application Code Running on CME

1.2.2 Host-based Execution of User Code

When located on a host controller the user code communicates via one of the available host interfaces to the PMD controller. Depending on the PMD product being used and the connection type chosen by the user this may be serial, CANbus, Ethernet, SPI (Serial Peripheral Interface), or PC/104. The format of these communications is one of two packet based protocols, either PRP, which is short for PMD Resource Access Protocol, or Magellan protocol. The user need not be concerned however with the packet format because these details are handled automatically when code is written in C-Motion.

While some systems will plan from the beginning to execute code on the host, other systems may execute the user code on the host during code development and then transition to execution on a C-Motion Engine. The advantage of developing the user application code on the host is that the user has access to the PC's keyboard, mouse, or touch screen user interface facilities, as well as, often, access to advanced software development and monitoring tools. The choice of software tools to compile and debug the C-Motion code when run on the host is determined by the user.

For more information on the Magellan packet protocol refer to the *C-Motion Magellan Programming Reference*. For more information on the PRP protocol refer to the either *C-Motion/PRP Programming Reference* or the *C-Motion/PRP II Programming Reference*.

1.2.3 C-Motion Engine-based Execution of User Code

When located on the PMD controller's C-Motion Engine the user code communicates directly to the resources available on the controller such as the Magellan Motion Control IC. This has speed advantages both in communicating with those resources, and in real time code execution predictability. The software tools used to compile and debug C-Motion code when run on the C-Motion Engine are provided by PMD and are contained in C-Motion SDKs (Software Development Kits).

Executing the code directly on the controller allows the controller to function as a fully standalone controller. In this mode a host controller network communication link is not needed, and one or more of the PMD controller's communication ports or digital I/O ports are typically used to interface to user-operated buttons or a touch screen. This is also a common configuration for machines that will interface to a central PLC (Programmable Logic Controller).

Alternatively, code can be executed on the PMD controller but be programmed in such a way as to receive or send commands from a host network, thus forming a local controller within a larger system. For example for a device with a C-Motion Engine controlling a three-axis gantry a host may send high level commands to it such as "move the gantry to location X, Y, Z". The user code executing on the C-Motion Engine parses these incoming commands and generates whatever axis-specific motions (and other functions) are required from each of the three controlled motion axes to execute the high level host command.

1.2.4 C-Motion Resource Access Virtualization

C-Motion source code can be written once, and then compiled to execute on either a host or on the C-Motion Engine of the PMD controller. This is a very powerful feature of C-Motion which makes it easy to develop and test user application code on the host PC and then execute the code directly on the C-Motion Engine in the production application.

What makes this possible is the ability of C-Motion to virtualize access to the PMD controller's resources. When the user application source code is compiled for the host PC, C-Motion commands automatically result in PRP packets being sent via the host interface. When compiled for the C-Motion Engine, the same C-Motion source code results in an executable program being created that communicates directly to the resources on the controller.

Note that one exception to this is features of the C-Motion Engine operating system which have no equivalents, or different equivalents, when running on the PC. Examples of features such as this include mailboxes and local digital input events. In addition execution speed of the C-Motion application code may vary significantly as it may access clock/timing-based functions**.**

1.2.5 C-Motion SDKs

There are three different C-Motion SDKs; C-Motion Magellan, C-Motion PRP, and and C-Motion PRP II. All of these SDKs are available from the PMD website. For detailed information on C-Motion PRP refer to the *C-Motion PRP Programming Reference*. Here is more information on each:

- **C-Motion Magellan SDK** – an SDK (Software Developer Kit) for creating motion applications using the C/C++ programming language for PMD products that utilize a direct Magellan or Juno formatted protocol. Note that this SDK should not be used for C-Motion applications that will be run on C-Motion Engine.

- **C-Motion PRP SDK** – an SDK for creating PC and downloadable user code for systems utilizing either a PRP (PMD Resource Access Protocol) protocol device or a Magellan/Juno protocol device. C-Motion PRP is also used in motion applications that will use the .NET (C#, VB) programming languages.
- **C-Motion PRP II SDK** – This SDK is similar to C-Motion PRP but is used with ION/CME N-Series ION Digital Drives. Compared to standard C-Motion PRP, C-Motion PRP II supports additional features such as multi-tasking, mailboxes, mutexes, and enhanced event management.

For detailed information on Magellan/Juno protocol C-Motion refer to the *C-Motion Magellan Programming Reference*. For detailed information on PRP protocol refer to the *C-Motion PRP Programming Reference* or the *C-Motion PRP II Programming Reference*.

1.2.6 C-Motion Versions

For reference the following table shows the C-Motion versions that can be used with each PMD product family:

Product Family	Compatible C-Motion Versions
Magellan ICs	C-Motion Magellan, C-Motion PRP*
Juno ICs	C-Motion Magellan, C-Motion PRP*
ION/CME N-Series	C-Motion PRP II
ION 500	C-Motion Magellan, C-Motion PRP*
ION/CME 500	C-Motion PRP
ION 3000	C-Motion Magellan, C-Motion PRP*
Prodigy PC/104	C-Motion Magellan, C-Motion PRP*
Prodigy/CME PC/104	C-Motion PRP
Prodigy/CME Stand-Alone	C-Motion PRP
Prodigy/CME Machine-Controller	C-Motion PRP

*C-Motion PRP typically only used for .NET support, or if a mix of Magellan/Juno protocol and PRP protocol devices are attached.

1.3 C-Motion Engine User Code Development Overview

The next few sections provide an overview of the process used to create, compile, download, and run C-Motion application code on the C-Motion Engine (CME).

1.3.1 Connections Overview

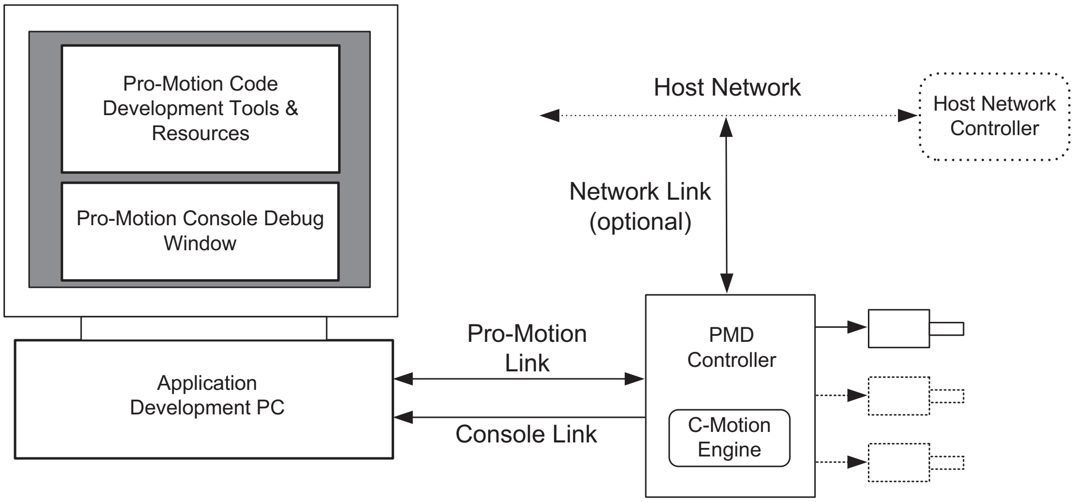


Figure 1-2: Typical Connection Links During C-Motion Engine Code Development

Figure 1-2 shows a typical connection architecture for C-Motion Engine user code development. There are two or three separate communication links depending on the application:

The **Pro-Motion link** connects the PC that holds Pro-Motion, the user source code, and the source code editing and compiling tools to the PMD controller that will receive the user code. This link allows the user code, once compiled and converted into a .bin file format, to be downloaded and eventually executed on the C-Motion Engine. In addition this link can be used to monitor the status of the motion system even while the CME-based user application code is executing. This may be useful, for example, to track the location and status of the controlled axis.

The Pro-Motion link uses PRP (PMD Resource Access Protocol) to communicate with the PMD controller.

The Pro-Motion link is generally always present during CME code development and execution.

The second link in a typical CME code development setup is the **Console link**. The console channel provides a convenient pathway for sending printf type statements from the user code running on the PMD controller to a separate monitor or to the Pro-Motion debug console window.

The console link does not have a protocol as such and transmits in ASCII whatever messages are sent using printf commands executed in the CME user code. In cases where a separate console link is not possible or not supported such as the CAN interface, the PRP console interface can be used (N-Series ION only). When the PRP console interface is used console output is buffered internally and extracted over the Pro-Motion link via PRP protocol.

While a console port connection may be useful, particularly in the earlier stages of user code development, not all systems will need or use a console channel link.

In a particular application many other types of connections which are used in the overall application may exist but which are not directly related to the functions needed for Pro-Motion or Console channels. Examples include controller-attached peripherals such as a RS-232 link to a hand held interface device.

Another common connection type is called a **Network Link**. Network links are optional and used in systems that process commands from a host network and then interpret these commands in the PMD controller.

> These various network links may be physically separate communication links or they may be separate virtual channels (port IDs) on a communications link such as Ethernet or CANbus.

1.3.2 Typical Code Development Session

With the CME code development connections in place you are ready to start building your application using the tools provided in the C-Motion engine development system. These C-Motion engine development tools are used to create/ edit, compile, and download user application code into the PMD controller.

The development system can download the file image for the current code project being worked on, or a specific named file can be downloaded. Downloaded files images end with a ".bin" extension. Only one code image file may be downloaded into the C-Motion Engine at a time. Downloading a new image automatically erases the previous code image.

There are times when it may be useful to read specific characteristics of a code file that has been downloaded into the C-Motion Engine. For example a host controller in a production environment may want to confirm that the host application code version actually loaded on the C-Motion Engine matches the expected production code version. To confirm this information Pro-Motion displays, in the C-Motion Engine Window, the file name of the downloaded user application code, the checksum of the downloaded file, the date & time of file creation, and the version number of the C-Motion Engine itself is displayed.

1.3.2.1 Code Compilation, Downloading & Execution

Once a .bin file has been generated it can be transferred and stored in the C-Motion Engine using Pro-Motion. After the user code has been loaded into the C-Motion Engine unit it is ready to be executed. If the auto-execution flag has been set then the simplest way to begin executing the code is to power cycle the PMD controller unit. After initialization the user code will execute automatically.

If code execution is set to manual, with the the PMD controller unit powered on the code can be started using Pro-Motion via the C-Motion Engine Window. Once the code executes you should begin to see any printf statements that you have embedded in the code displayed in the console channel. During initial code development this is a convenient method to confirm that the code loaded correctly and is executing.

To exercise and become familiar with the C-Motion Development tools Section 1.4, "Example Code Development Tutorial," provides a tutorial to compile, link, and download a sample "hello world" application included in the SDK. When executed, this simple "hello world" code sends a "hello world" message to the console. As a next step you can edit the source code file, for example changing the output message slightly, and then re-compile, download, and execute this modified .bin code image.

1.4 Example Code Development Tutorial

The next several sections of this chapter provide an example code development session that illustrates the basic elements of the CME Development Tools and provides a foundation for developing your own application for execution on a C-Motion Engine.

Here is a summary of the steps that will comprise this C-Motion Engine Development Tools 'getting started' tutorial:

1 If not yet installed, install the SDK appropriate for the product you are using.

2 Set up host communications for your tutorial session.

3 Load the Hello project, and compile and link this example project. Two different sets of instructions are provided to accomplish this depending on the version of the C-Motion SDK you are using. For ION/ CME N-Series Digital Drive users who installed C-Motion PRP II follow the instructions in Section 1.4.3.1, "Edit and Build with User-Provided Tools." For all other ION/CME or Prodigy/CME product follow the instructions in Section 1.4.3.2, "Edit and Build with Programmer's Notepad."

4 Launch Pro-Motion, and download the Hello World project binary file to the CME device's C-Motion Engine.

5 Begin C-Motion Engine code execution, and monitor results on the console window.

Once these steps have been accomplished, the tutorial is complete.

1.4.1 Step #1, SDK Installation

By the time you begin code development you will have already downloaded software in connection with setting up a PMD product developer kit. Therefore, depending on what was installed from that original DK download the instructions below may not be needed at all, or may only be needed in part.

In any case, for the sake of completeness these instructions are written assuming no PMD software download has yet occurred.

The software distribution is downloaded from the PMD website at the URL: https://www.pmdcorp.com/resources/software.

All software applications are designed to work with Microsoft Windows.

To install the software:

1 Go to the Software Downloads section of PMD's website located at: https://www.pmdcorp.com/resources/software and select the download for "ION/CME N-Series Digital Drive Developer Kit Software" if using an N-Series ION, and select "Developer Kit Software" if using any other PMD product.

2 After selecting download you will be prompted to register your DK, providing the serial # for the DK and other information about you and your motion application.

3 After selecting submit the next screen will provide a link to the software download. The software download is a zip file containing various installation programs. Select this link and downloading will begin.

4 Once the download is complete extract the zip file and execute an install of Pro-Motion. After Pro-Motion install is complete N-Series ION users should download the C-Motion PRP II SDK. All other PMD DK users should download the C-Motion PRP SDK.

1.4.2 Step #2, Host Communication Setup

The next step of the tutorial is to set up the host and console connections. The following table shows the Pro-Motion and console connectivity that the device should be set to while using this tutorial for various types of CME devices:

CME device type	Pro-Motion connectivity to PC	Console Interface
N-Series ION, Ethernet host type	Ethernet	UDP
N-Series ION, Serial or CAN/SPI host type	3-pin programming port	PRP
Prodigy/CME Machine Controller	Ethernet	UDP
Prodigy/CME Stand-Alone	Ethernet	UDP
Prodigy/CME PC/104	Ethernet	UDP
ION/CME 500	Ethernet	UDP

Although it is possible to set the Pro-Motion communication port (the port that will send and receive PRP messages to/from the CME device), and the console port (the destination of printf and similar messages sent from the C-Motion Engine) to values other than those indicated in the table above, for simplicity this getting started tutorial will assume these connections.

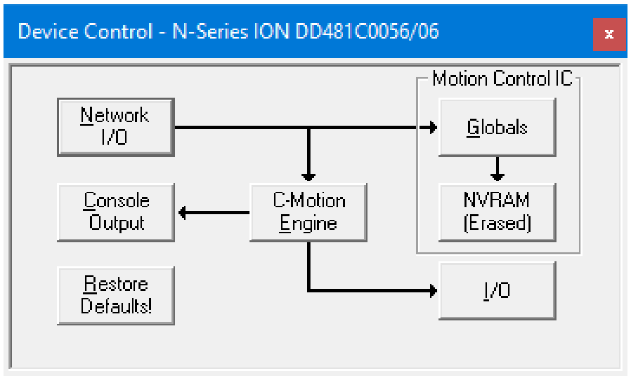


Figure 1-3: Pro-Motion Device Control Window

Setting the Pro-Motion Link: If your existing Pro-Motion communication link is different than the one in the table you should re-connect Pro-Motion with the PMD controller accordingly. The developer kit user manual for the product you are using should provide detailed instructions on how to do this.

Setting the Console link: To set the console connection type use the Console Output button located in the Device Control window. Figure 1-3 shows an image of this dialog box.

1.4.3 Step #3, Build the Project

In this step we will compile and link the Hello World project example.

Hello.c is a simple application intended for execution on the CME device's C-Motion Engine that:

- Sends "Hello, world" to the console window.
- Sends the current encoder position of axis #1 of the Magellan Motion Control IC to the console window.
- Sends updated values of the actual position, once per second, if the actual encoder position changes.

Depending on what PMD Controller product you are using you should use one or two different sets of instructions to accomplish this:

ION/CME N-Series Digital Drive users should follow the instructions beginning at Section 1.4.3.1, "Edit and Build with User-Provided Tools."

All other ION/CME or Prodigy/CME product users should follow the instructions beginning at Section 1.4.3.2, "Edit and Build with Programmer's Notepad."

1.4.3.1 Edit and Build with User-Provided Tools

Any text editor can be used to edit the example project files. Building the project is performed using the build.bat file located in each project directory.

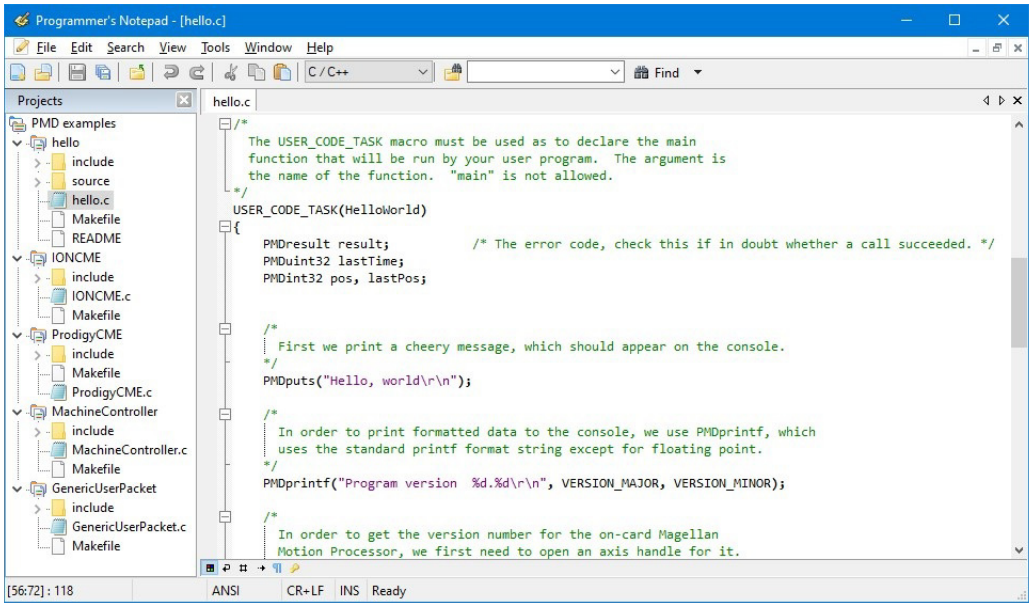
Alternatively, a Microsoft Visual Studio project file (.vcxproj) is also provided to aid in editing and compiling. For example, the MSVC project file for the Hello World project is located here:

<dir where zip file has been extracted>\CMESDK\CMECode\Examples\Hello\Hello.vcxproj

When ready skip to Section 1.4.4, "Step #4, Downloading the Code," for a description of the next tutorial step.

1.4.3.2 Edit and Build with Programmer's Notepad

The CMECode examples in the SDK for the Prodigy/CME and ION/CME products contain project files for use with Programmer's Notepad. Programmer's Notepad can be downloaded from http://www.pnotepad.org/download/. An example session for the Hello World example is shown in Figure 1-4.



```
/*
  The USER_CODE_TASK macro must be used as to declare the main
  function that will be run by your user program.  The argument is
  the name of the function.  "main" is not allowed.
*/
USER_CODE_TASK(HelloWorld)
{
    PMDresult result;           /* The error code, check this if in doubt whether a call succeeded. */
    PMDuint32 lastTime;
    PMDint32 pos, lastPos;


    /*
     First we print a cheery message, which should appear on the console.
    */
    PMDputs("Hello, world\r\n");

    /*
     In order to print formatted data to the console, we use PMDprintf, which
     uses the standard printf format string except for floating point.
    */
    PMDprintf("Program version  %d.%d\r\n", VERSION_MAJOR, VERSION_MINOR);

    /*
     In order to get the version number for the on-card Magellan
     Motion Processor, we first need to open an axis handle for it.
```

Figure 1-4: Programmer's Notepad with Example Program Shown

Each project can be compiled by typing “make” in the associated project folder. For example, to compile the Hello World example go to the

“C:\Users\<username>\Documents\PMD\CMESDK\CMECode\Examples\Hello” folder and type make.

1.4.4 Step #4, Downloading the Code

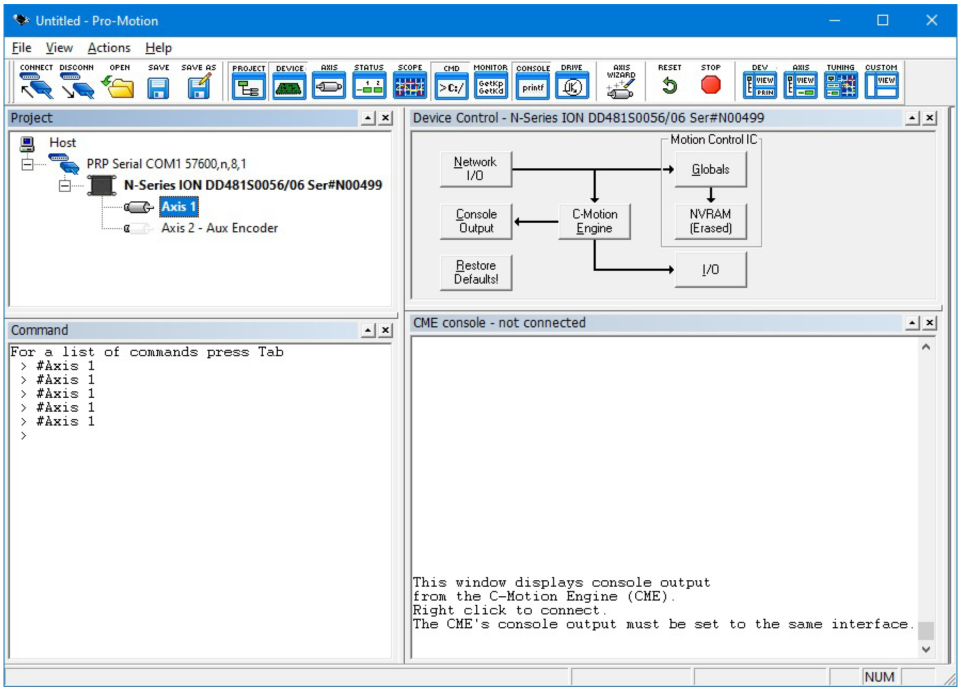
Once the.bin file has been created, the remaining steps of the development process will occur via Pro-Motion.

In any case, the following steps are used to continue the tutorial process:

1 Locate the Pro-Motion icon from the PC desktop and launch.

Depending on how Pro-Motion was last used, you may need to click the '**DEV**.' (short for development view) icon on the tool bar at the top, right of the application window. Figure 1-5 shows a typical screen view with Pro-Motion in this view mode.

Figure 1-5: Pro-Motion Starting Screen in Dev. View



```
For a list of commands press Tab
> #Axis 1
> #Axis 1
> #Axis 1
> #Axis 1
> #Axis 1
>
```

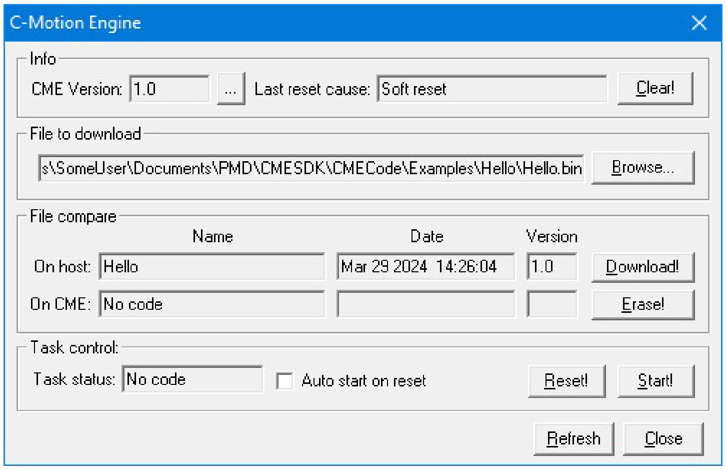
```
This window displays console output
from the C-Motion Engine (CME).
Right click to connect.
The CME's console output must be set to the same interface.
```

At the center of the screen you will see the **Device Control** window. This convenient graphical representation of the CME device allows you to manage card-level functions. The **Console** window is to the bottom. This will show messages output from the C-Motion Engine.

- Click the **C-Motion Engine** button.

A dialog box as shown in Figure 1-6 appears providing various status and version information for the C-Motion Engine and information on any user application code file (.bin files) that may be loaded into it.

Figure 1-6: C-Motion Engine Dialog Box



CME Version: 1.0

Last reset cause: Soft reset

s\SomeUser\Documents\PMD\CMESDK\CMECode\Examples\Hello\Hello.bin

	Name	Date	Version
On host:	Hello	Mar 29 2024 14:26:04	1.0
On CME:	No code		

Task status: No code

□ Auto start on reset

2 In the C-Motion Engine window, use the **File To Download** window or the **Browse** button to locate the "**Hello.bin**" file (the examples are typically located in the folder "**Documents\PMD\CMESDK\CMECode**").

3 Click the **Download** button, and the download process begins.

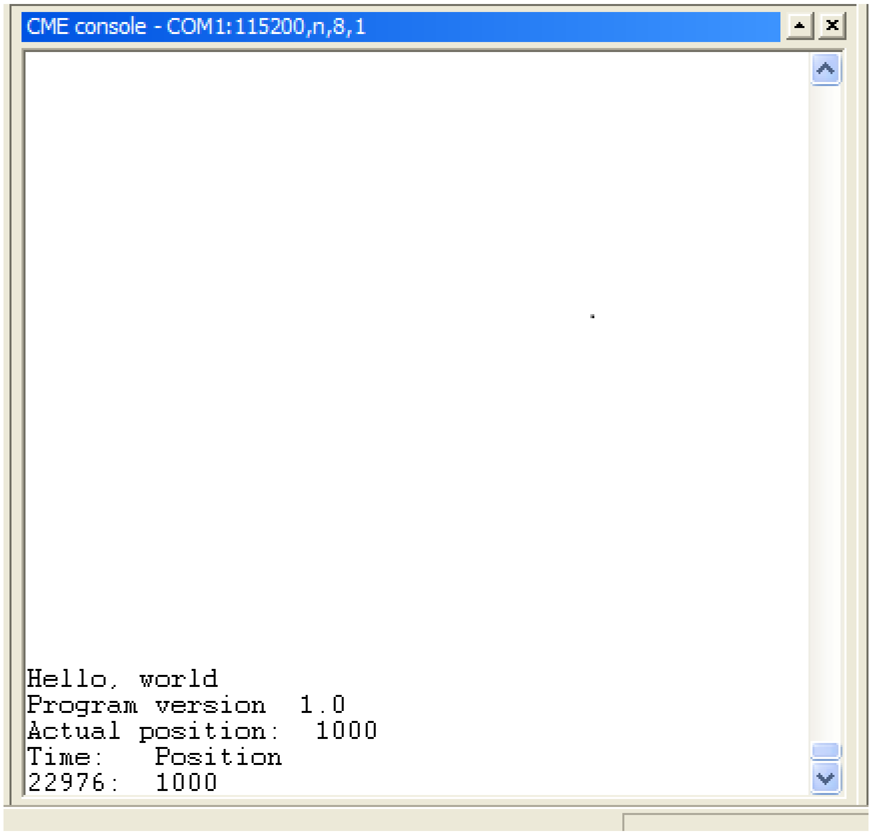
4 After a successful flash download the name, user version number, and checksum of the program is displayed in the **C-Motion Engine** window.

5 An additional dialog box asks you whether you would like to begin executing the code.

1.4.5 Step #5, Executing the Code

Upon successfully downloading the Hello.bin file in the steps above, the code is loaded into non-volatile memory in the C-Motion Engine and ready to execute. The following instructions complete the process of executing and monitoring code execution on the C-Motion Engine.

1 From step 4 above, click **Start** to begin code execution. As shown in Figure 1-7 you should immediately see a number of console screen messages appear in the window to the bottom of the Pro-Motion screen.



```
Hello, world
Program version  1.0
Actual position:  1000
Time:   Position
22976:  1000
```

Figure 1-7: Pro-Motion Console Window

These message were sent by the C-Motion Engine to the **Console** window as it executed the downloaded code. Console messages are special communications that use the standard output stream (stdio) to display messages from the C-Motion Engine.

2 If an encoder is connected to axis #1 of your CME device, turn the encoder and updated actual position values will be output to the **Console** window.

This program continues executing indefinitely until you either load another .bin file to the C-Motion Engine, or reset the card.

Congratulations! You have completed the basic 'Hello, world' tutorial and can now continue code development by creating your own projects or expanding on the example projects included with your SDK.

2. Application Development Tools

In This Chapter

- Example projects
- Pro-Motion
- Software Development

2.1 Example projects

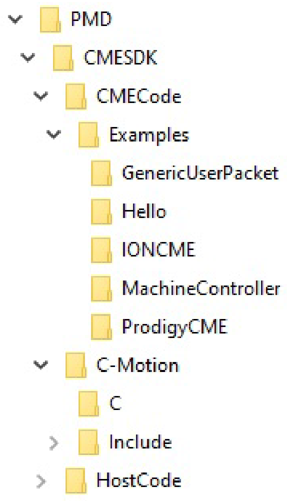
There are some example projects included with the C-Motion Engine Development Tools SDK. Each example project can be compiled by typing “make” (or “build” for N-ION/CME) in the associated project folder.

For example, to compile the Hello World example go to the

“C:\Users\<username>\Documents\PMD\CMESDK\CMECode\Examples\Hello” folder and type make.

Each example may include a README file, which describes what the example does and any configuration or special hardware that might be required.

The examples can be compiled as is or can be used as the basis of a customized system through further modification.



```
v PMD
  v CMESDK
    v CMECode
      v Examples
          GenericUserPacket
          Hello
          IONCME
          MachineController
          ProdigyCME
    v C-Motion
        C
      > Include
    > HostCode
```

Figure 2-1: Directory Tree of the Examples Folder

> The examples provided with the C-Motion Development Tools SDK are intended purely for illustration purposes. They are not warranted to be error free, nor are they warranted to be suitable for use in a particular application. It is up to the user to insure the quality of their developed code.

2.2 Pro-Motion

Pro-Motion is a software prototyping and control tool used for all Magellan family products. Most Pro-Motion features are described in the *Pro-Motion User's Guide*, but features used only with CME devices are described here.

When used with a CME device, Pro-Motion provides a Device Control pane, which may be turned on and off with the device icon in the left half of the toolbar. The right half of the toolbar includes a Dev (short for development) icon which may be used to set Pro-Motion to a view that includes the Device Control pane, and other panes useful for CME code development.

2.2.1 The Console

The CME has the notion of a *console,* a peripheral used for displaying messages from C-Motion Engine user programs. This is useful for debugging, reporting progress, and sanity checking.

For Prodigy/CME boards the default output console channel is set to none. For ION/CME N-Series Driver the default console output channel is PRP.

To change where the CME device outputs its console information the Console Output box in the Device Control pane is used. By clicking on this box, you can select one of the options: serial, Ethernet (UDP/IP), PRP, or none.

Changing these values changes the power up default console settings only, so you must reset the card for these changes to take effect.

To enable Pro-Motion displaying messages on the console window, right-click in the CME Console window and select the console channel that matches what the card uses for console output.

After reset, if a user program is running, then the console shows the output of any PMDprintf, PMDputs, or PMDputch calls.

2.2.2 Downloading and Running User Programs

The C-Motion Engine in a CME device may be programmed with a single user program at a time, and downloading may be done using the C-Motion Engine box in the Device Control pane.

2.3 Software Development

2.3.1 CME Development Software Libraries

C-Motion Engine user programs may use standard C library and math library calls, which are provided by the Red Hat newlib C library, which is intended specifically for embedded applications. For N-Series ION floating point support is provided in hardware. For all other CME products floating point support is provided in software, so there is a definite performance penalty for floating point arithmetic when compared to fixed point.

Dynamic memory allocation is supported, using the standard malloc and free calls. For N-Series ION approximately 500 KB of heap space is available. For all other CME products approximately 7 kilobytes of heap space are available. Because of the possibilities of heap fragmentation and memory leaks in application code it is strongly recommended that dynamic allocation be kept as static as possible, for example by allocating all necessary data structures at initialization time.

Standard I/O functions are supported only for the standard output streams stdout and stderr, which are connected to the console. Because there are no file systems available to CME device user programs there is no way to open a file.

2.3.2 PMD Software Libraries

Almost all operations performed by C-Motion Engine user programs that deal with physical devices and resources are accomplished using PMD C language libraries.

It may not be obvious at the outset of development exactly which parts of an application should run in the CME and which on a host computer. In order to make it simpler to change this division of labor, and to allow prototyping of CME code on a PC, almost all of the PMD library call sequences are the same in both host-based and C-Motion Engine environments, although the internal details of implementation and data structures differ substantially.

There are several categories of physical device that CME user programs may deal with:

- Magellan Motion Processors, either the CME device processor, or Magellan attached devices, such as non-CME ION modules or non-CME Prodigy cards. These are controlled using the PMD C-Motion library, which is documented in the *C-Motion Magellan Programming Reference*. C-Motion uses a representation of a Motion Processor control axis called an axis handle for all commands. Although the means of obtaining the axis handle are different for the various Magellan devices, the use of the axis handle once obtained is identical.
- Peripheral connections, meaning communication links to off-card devices, using serial, Ethernet, CANBus, SPI, or PC/104 bus channels. Peripheral handling is supported by a PMD C language library which is used only with CME devices and documented in the *C-Motion PRP Programming Reference*. Peripheral connections are represented by a simple peripheral handle object, which may be read or written. There are special procedures for using peripheral connections to address Magellan attached devices or other CME devices.
- Other on-card resources, such as dual-ported RAM, and general purpose digital I/O pins, are accessed either by using the Prodigy/CME library or C-Motion calls for non-Magellan resources documented in the *Magellan Motion Processor Programmer's Reference.*

2.3.3 User Code Architecture

A CME program is normally in the form of a non-terminating loop, because it must continue doing its job as long as the CME device is powered. If it is necessary to exit because of an error condition the PMDTaskAbort procedure should be used.

The C-Motion Engine is a specialized computer that does almost nothing but motion control. This means that a CME user program may freely use, and rely upon, almost all the resources (memory, processor cycles) available to the CME, but it also means that the programmer is responsible for not exceeding the limits on any resource. These resources include:

- Stack space – deeply nested procedure calls, and large data structures in auto storage should be avoided.
- Processor cycles – the developer should use timing procedures such as PMDGetTickCount to verify that each part of an application can run within its time budget. Handling peripheral input and output also requires processor cycles, so heavy network traffic, particularly Ethernet traffic, may overload the processors.
- Data space plus heap space is limited to approximately 500 kilobytes for N-Series ION and 7 kilobytes for all other CME products. It is suggested that all dynamic allocation be done in the initialization phase of a user program, so that unexpected allocation failures due to memory leaks or heap fragmentation do not cause problems.
- Code space is limited to approximately 256 kilobytes. Fortunately code size limitations can be found at build time. One of the best ways of reducing code size is to factor code so that the same procedure performs its task in several places. Cut and paste coding bloats code size.

CME programs require a small amount of boiler plate, specifically the USER_CODE_VERSION and USER_CODE_TASK macros. USER_CODE_VERSION is used to specify major and minor version numbers for the user program, but is required even if you don't care about versioning. It must be used once, at top level (outside any procedure body). USER_CODE_TASK is used to define the procedure used as the main entry point in the user program. A skeleton user program might look like this:

```
#include "C-Motion.h"      // C-Motion library calls, for Motion Processor control
#include "PMDsys.h"        // Operating-system specific procedures

#define MAJOR_VERSION 0 // These are for the user, and may be freely chosen.
#define MINOR_VERSION 9

USER_CODE_VERSION(MAJOR_VERSION, MINOR_VERSION) // Required

USER_CODE_TASK(myjob)
{
   // Get ready to do something useful here.
    while (!0) { …
        // Do something useful here.
  }
}
```

The first tool for discovering what your user program is doing, and why it might be failing, is the console. Information on progress, status, and debugging values may be printed using PMDprintf, PMDputs, and PMDputch. for non-N-Series ION products PMDprintf may not be used to format floating point numbers, instead sprintf using a temporary buffer must be used.

Another useful diagnostic tool is the set of general purpose digital I/O. With the use of an oscilloscope these pins can send diagnostic output much faster than the console procedures.

CME user programs may cause processor exceptions, for example, by reading from or writing to illegal addresses, or by trying to execute illegal instructions, probably because of stack corruption or bad function pointers. An exception will cause a reset of the entire CME device, because it is impossible to know what data structures were corrupted before the exception occurred. After the reset some diagnostic information, including the code address of the exception, register values at the time, and some stack contents, will be printed to the console. This information may be useful in determining where the program went wrong.

For non-N-Series ION products C-Motion Engine user program has a single thread of control; multi-threading or interrupt handling are not available. Because of this restriction CME user programs have a very simple model, and semaphores or other locking mechanisms are not used, which means that almost everything done by a user program must be driven by polled input, with or without timeouts. The recommended way of structuring a CME user program is as a state machine, which uses one or more variables to represent an enumerated set of states, and transitions from one state to the next in a way determined by the inputs read in each state.

Appendix A. - Copyright Notices

A.1 Overview

The following sections provide required copyright notifications for the open source software libraries, LWIP (Lightweight TCP/IP Stack), and Newlib, that PMD uses as part of its C-Motion Engine Development Tools. If you have any questions on these copyright notices please contact PMD for more information.

A.2 Lightweight TCP/IP Stack (LWIP)

lwIP is licensed under the BSD license:

A.3 Newlib

The newlib subdirectory is a collection of software from several sources.

Each file may have its own copyright/license that is embedded in the source file. Unless otherwise noted in the body of the source file(s), the following copyright notices will apply to the contents of the newlib subdirectory:

(1) Red Hat Incorporated

(2) University of California, Berkeley

(3) David M. Gay (AT&T 1991, Lucent 1998)

The author of this software is David M. Gay.

Copyright (c) 1991 by AT&T.

Permission to use, copy, modify, and distribute this software for any purpose without fee is hereby granted, provided that this entire notice is included in all copies of any software which is or includes a copy or modification of this software and in all copies of the supporting documentation for such software.

THIS SOFTWARE IS BEING PROVIDED "AS IS", WITHOUT ANY EXPRESS OR IMPLIED WARRANTY. IN PARTICULAR, NEITHER THE AUTHOR NOR AT&T MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND CONCERNING THE MERCHANTABILITY OF THIS SOFTWARE OR ITS FITNESS FOR ANY PARTICULAR PURPOSE.

The author of this software is David M. Gay.

Copyright (C) 1998-2001 by Lucent Technologies

All Rights Reserved

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that the copyright notice and this permission notice and warranty disclaimer appear in supporting documentation, and that the name of Lucent or any of its entities not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

LUCENT DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL LUCENT OR ANY OF ITS ENTITIES BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

(4) Advanced Micro Devices

Copyright 1989, 1990 Advanced Micro Devices, Inc.

This software is the property of Advanced Micro Devices, Inc (AMD) which specifically grants the user the right to modify, use and distribute this software provided this notice is not removed or altered. All other rights are reserved by AMD.

AMD MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS SOFTWARE. IN NO EVENT SHALL AMD BE LIABLE FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH OR ARISING FROM THE FURNISHING, PERFORMANCE, OR USE OF THIS SOFTWARE.

So that all may benefit from your experience, please report any problems or suggestions about this software to the 29K Technical Support Center at 800-29-29-AMD (800-292-9263) in the USA, or 0800-89-1131 in the UK, or 0031-11-1129 in Japan, toll free. The direct dial number is 512-462-4118.

Advanced Micro Devices, Inc.

29K Support Products

Mail Stop 573

5900 E. Ben White Blvd.

Austin, TX 78741

800-292-9263

(5) C.W. Sandmann

Copyright (C) 1993 C.W. Sandmann

This file may be freely distributed as long as the author's name remains.

(6) Eric Backus

(C) Copyright 1992 Eric Backus

This software may be used freely so long as this copyright notice is left intact. There is no warrantee on this software.

(7) Sun Microsystems

Copyright (C) 1993 by Sun Microsystems, Inc. All rights reserved.

Developed at SunPro, a Sun Microsystems, Inc. business.

Permission to use, copy, modify, and distribute this software is freely granted, provided that this notice is preserved.

(8) Hewlett Packard

(c) Copyright 1986 HEWLETT-PACKARD COMPANY

To anyone who acknowledges that this file is provided "AS IS" without any express or implied warranty: permission to use, copy, modify, and distribute this file for any purpose is hereby granted without fee, provided that the above copyright notice and this notice appears in all copies, and that the name of Hewlett-Packard Company not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

Hewlett-Packard Company makes no representations about the suitability of this software for any purpose.

(9) Hans-Peter Nilsson

Copyright (C) 2001 Hans-Peter Nilsson

Permission to use, copy, modify, and distribute this software is freely granted, provided that the above copyright notice, this notice and the following disclaimer are preserved with no changes.

THIS SOFTWARE IS PROVIDED ``AS IS'' AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

(10) Stephane Carrez (m68hc11-elf/m68hc12-elf targets only)

Copyright (C) 1999, 2000, 2001, 2002 Stephane Carrez (stcarrez@nerim.fr)

The authors hereby grant permission to use, copy, modify, distribute, and license this software and its documentation for any purpose, provided that existing copyright notices are retained in all copies and that this notice is included verbatim in any distributions. No written agreement, license, or royalty fee is required for any of the authorized uses.

Modifications to this software may be copyrighted by their authors and need not follow the licensing terms described here, provided that the new terms are clearly indicated on the first page of each file where they apply.

(11) Christopher G. Demetriou

Copyright (c) 2001 Christopher G. Demetriou

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The name of the author may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

(12) SuperH, Inc.

Copyright 2002 SuperH, Inc. All rights reserved

This software is the property of SuperH, Inc (SuperH) which specifically grants the user the right to modify, use and distribute this software provided this notice is not removed or altered. All other rights are reserved by SuperH.

SUPERH MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS SOFTWARE. IN NO EVENT SHALL SUPERH BE LIABLE FOR INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH OR ARISING FROM THE FURNISHING, PERFORMANCE, OR USE OF THIS SOFTWARE.

So that all may benefit from your experience, please report any problems or suggestions about this software to the SuperH Support Center via e-mail at softwaresupport@superh.com .

SuperH, Inc.

405 River Oaks Parkway

San Jose

CA 95134

USA

(13) Royal Institute of Technology

Copyright (c) 1999 Kungliga Tekniska HÃ¶gskolan

(Royal Institute of Technology, Stockholm, Sweden).

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. . Neither the name of KTH nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY KTH AND ITS CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL KTH OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR

BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

(14) Alexey Zelkin

Copyright (c) 2000, 2001 Alexey Zelkin <phantom@FreeBSD.org>

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

(15) Andrey A. Chernov

Copyright (C) 1997 by Andrey A. Chernov, Moscow, Russia.

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF

MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

(16) FreeBSD

Copyright (c) 1997-2002 FreeBSD Project.

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

(17) S. L. Moshier

Author: S. L. Moshier.

Copyright (c) 1984,2000 S.L. Moshier

Permission to use, copy, modify, and distribute this software for any purpose without fee is hereby granted, provided that this entire notice is included in all copies of any software which is or includes a copy or modification of this software and in all copies of the supporting documentation for such software.

(18) Citrus Project

(19) Todd C. Miller

(20) DJ Delorie (i386)

(21) Mike Barcroft

IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

(22) Konstantin Chuguev (--enable-newlib-iconv)

Copyright (c) 1999, 2000

Konstantin Chuguev. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1 Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2 Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

iconv (Charset Conversion Library) v2.0

(23) Artem Bityuckiy (--enable-newlib-iconv)

Copyright (c) 2003, Artem B. Bityuckiy, SoftMine Corporation.

Rights transferred to Franklin Electronic Publishers.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1 Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2 Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

(24) - Alex Tatmanjants (targets using libc/posix)

Copyright (c) 1995 Alex Tatmanjants <alex@elvisti.kiev.ua>

at Electronni Visti IA, Kiev, Ukraine.

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1 Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2 Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

(25) - M. Warner Losh (targets using libc/posix)

Copyright (c) 1998, M. Warner Losh <imp@freebsd.org>

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1 Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2 Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

(26) - Andrey A. Chernov (targets using libc/posix)

Copyright (C) 1996 by Andrey A. Chernov, Moscow, Russia.

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1 Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2 Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

(27) - Daniel Eischen (targets using libc/posix)

For additional information, or for technical assistance, please contact PMD at (978) 266-1210.

You may also e-mail your request to support@pmdcorp.com

Visit our website at http://www.pmdcorp.com

PMD

Performance Motion Devices
80 Central Street.
Boxborough, MA 01719