

# **KwikPeg<sup>TM</sup>**

## **Graphical User Interface**

### **USER'S GUIDE**

**First Printing: April 15, 2000**  
**Last Printing: June 15, 2003**

**Copyright © 2000-2003**

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



## TECHNICAL SUPPORT

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

Engineering Change Notices (ECNs) are provided periodically to repair faults or to improve performance. You will automatically receive these updates for a period of one year. After that period, you may purchase additional updates. Please keep us informed of the primary user in your company to whom these update notices and other pertinent information should be directed.

Should you require direct technical assistance in your use of this KADAK software product, engineering support is available by telephone, fax or e-mail without charge. KADAK reserves the right to charge for technical support services which it deems to be beyond the normal scope of technical support.

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

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

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

**Copyright © 2000-2003 by KADAK Products Ltd.  
All rights reserved.**

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

### **Copyright Acknowledgement**

The KwikPeg Graphical User Interface (GUI) is an enhanced version of the PEG Portable Embedded GUI created and distributed by Swell Software, Inc. Portions of this document have, with the kind permission of Swell Software, been derived from material which is the copyright of Swell Software, Inc.

### **DISCLAIMER**

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

### **TRADEMARKS**

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

# KwikPeg GUI User's Guide

## Table of Contents

	Page
<b>1. KwikPeg Overview</b>	<b>1</b>
1.1 Introduction.....	1
1.2 General Operation .....	6
KwikPeg Operation .....	8
Byte Ordering and Endianness.....	10
File System Use .....	10
1.3 KwikPeg Nomenclature.....	11
1.4 Memory Allocation Requirements .....	12
1.5 Debugging Aids .....	14
1.6 KwikPeg Sample Program - A Tutorial.....	15
<b>2. KwikPeg System Configuration</b>	<b>19</b>
2.1 Introduction.....	19
The KwikPeg Library .....	19
2.2 KwikPeg Configuration Builder .....	21
2.3 KwikPeg Library Parameter File .....	27
Target Parameters .....	28
OS Parameters.....	30
Drawing Parameters.....	33
Font Parameters .....	38
Image Conversion Parameters.....	40
Video Parameters .....	42
Input Device Parameters .....	44
File System Parameters.....	47
<b>3. KwikPeg System Construction</b>	<b>49</b>
3.1 Building an Application .....	49
3.2 Making the KwikPeg Library .....	50
3.3 Compiling Application Modules .....	60
3.4 Linking the Application .....	61
3.5 Integrating KwikPeg with AMX .....	62
3.5.1 AMX System Configuration .....	62
3.5.2 AMX Target Configuration.....	66
3.6 Constructing the KwikPeg Sample Program .....	68
3.7 An Alternate KwikPeg Library Building Method .....	70

## KwikPeg GUI User's Guide Appendices

	Page
<b>Appendix A. KwikPeg Glossary</b>	<b>A-1</b>
<b>Appendix B. KwikPeg Error Codes</b>	<b>B-1</b>
<b>Appendix C. KwikPeg Fonts</b>	<b>C-1</b>
C.1 KwikPeg Vector Fonts.....	C-1
C.2 The KwikPeg System Font .....	C-2
C.3 The KwikPeg Menu Font.....	C-3
 <b>Appendix D. KwikPeg Universal File System Interface</b>	 <b>D-1</b>
D.1 Introduction .....	D-1
D.2 KwikPeg File System Parameters.....	D-2
D.3 Using the AMX/FS File System.....	D-4
D.4 Using the MS-DOS File System.....	D-6
D.5 Using a Custom File System .....	D-7

## KwikPeg GUI User's Guide Table of Figures

	Page
Figure 1.2-1 KwikPeg Application Block Diagram .....	7
Figure 1.2-2 KwikPeg Operation.....	9
 Figure 2.1-1 Creating the KwikPeg Library Make File.....	 20
Figure 2.2-1 Configuration Manager Screen Layout.....	22
 Figure 3.2-1 KwikPeg Library Construction .....	 51
Figure 3.2-2 KwikPeg Compiler Configuration Header Files .....	53
Figure 3.2-3 KwikPeg Tailoring Files .....	55
Figure 3.7-1 Setting the KwikPeg Library Header File Path.....	71
 Figure C.1-1 KwikPeg Vector Font.....	 C-1
Figure C.2-1 The KwikPeg System Font.....	C-2
Figure C.3-1 The KwikPeg Menu Font .....	C-3

# 1. KwikPeg Overview

## 1.1 Introduction

The KwikPeg™ Graphical User Interface (GUI) is a compact, reliable, high performance tool, which enables you, the embedded system developer, to easily add graphics to your products. KwikPeg is an enhanced version of PEG™ (the Portable Embodied GUI), a professional, high-quality graphic system created by Swell Software, Inc.

The KwikPeg GUI includes a complete complement of drawing tools. You can readily tailor the KwikPeg GUI to accommodate your needs by using the KwikPeg Configuration Builder, a Windows® utility which makes configuring KwikPeg a snap. Your KwikPeg GUI will only include the features required by your application.

KwikPeg is ready for use with KADAK's AMX™ Mutitasking Kernel. KwikPeg is delivered to you ready for use on a particular target processor with any of the software development tools which KADAK supports for that target. You can concentrate on your application, knowing that the integration of KwikPeg with the AMX real-time operating system (RTOS) is fully functional. **No porting** is required.

The purpose of this manual is to provide the system designer and applications programmer with the information required to properly configure and implement a graphical user interface using the KwikPeg GUI. It is assumed that you are familiar with the architecture of your target processor. It is further assumed that you are familiar with the rudiments of microprocessor programming including the concepts of code, data and stack separation.

KwikPeg is available in C/C++ source format to ensure that regardless of your development environment, your ability to use and support KwikPeg is uninhibited. The source program may also include code fragments programmed in the assembly language of the target processor to improve execution speed.

The C and C++ programming languages, commonly used in real-time systems, are used throughout this manual to illustrate the features of KwikPeg.

## Installation

KwikPeg is delivered to you on a CD-ROM. During installation you will need your KwikPeg CD serial number and product installation key, both of which are recorded on a label attached to the CD-ROM case. The CD serial number is also printed on the face of the CD-ROM.

The CD serial number identifies your KwikPeg CD-ROM. The product installation key identifies the specific parts which you are entitled to install from the CD-ROM.

KwikPeg is installed by running the InstallShield® *SETUP.EXE* program located in the root directory of the CD-ROM. From the Windows Start, Run... menu, type *D:\SETUP.EXE* (where *D:* is your CD-ROM drive letter) and press Enter. Alternatively, browse the root directory of the CD-ROM and double click on the *SETUP.EXE* filename or icon.

The setup utility will lead you through the installation process. You may be requested to identify the software development tools which you plan to use for development. If so, only the tools specifically supported by KADAK will be listed.

The installation process will copy the product files into a directory of your choice on the disk drive of your choice. KwikPeg files will be installed in subdirectory *KPGnnn* where the number *nnn* comes from the KADAK part number PNnnn-1 used to identify KwikPeg. Throughout all KwikPeg manuals, the installation directory will be referred to as *KPGnnn*.

Note

The standard distribution of PEG for AMX as provided by Swell Software, Inc. must be installed in a directory named *PEG* before KwikPeg is installed.

## Manuals

This **KwikPeg GUI User's Guide** describes how the standard distribution of PEG from Swell Software, Inc. is used with the AMX RTOS.

The standard **PEG Programming Manual** from Swell Software, Inc. is provided in Adobe<sup>®</sup> Acrobat<sup>®</sup> PDF file format on the CD-ROM. This manual describes PEG, its methods of use and its programming methodology.

A separate on-line **KwikPeg Reference Manual** describes the complete KwikPeg C/C++ application programming interface. This manual is an extended version of the standard PEG Reference Manual provided by Swell Software, Inc., a copy of which is also included with PEG on the CD-ROM. View these HTML manuals with your browser.

### The KwikPeg GUI User's Guide

Chapter 1 provides an overview of the KwikPeg GUI. The general operation of KwikPeg is described and the nomenclature used by KADAK is introduced. Appendix A includes a glossary which will help when you are stuck trying to remember what one of the many GUI terms means. A number of topics unrelated to GUI issues are covered in this chapter. The use of KwikPeg in the AMX multitasking environment is described. KwikPeg memory allocation requirements are examined.

Chapter 2 is your system configuration guide. You may wish to read this chapter to learn how easy it is to use the KwikPeg Configuration Builder to customize the KwikPeg Library for your use.

Chapter 3 describes how KwikPeg is configured for use and combined with your application to form an executable load module. It also describes how to adapt your AMX configuration for use with KwikPeg.

Appendix D describes how to incorporate a file system so that KwikPeg services which require file access can be used.

## PEG Programming Manual

The PEG Programming Manual presents a look at KwikPeg internals and introduces basic concepts that are needed to fully understand how KwikPeg works. It describes the fundamental C/C++ data types and classes which you will use to create a graphical user interface. You will need to read and understand this material before you begin your GUI application software development.

The PEG Programming Manual also presents methods and examples which will prove invaluable when designing and creating your own GUI software. These examples and your actual GUI can be built and tested on a Windows workstation using the techniques introduced in this manual.

A discussion of the physical KwikPeg screen, the drawing tablet on which your GUI will be presented, is presented in the PEG Programming Manual. You will find descriptions of the various types of video and LCD controllers for which KwikPeg screen drivers are available. Input device drivers for keyboard and mouse are also described.

KwikPeg provides a number of software generation tools (Windows utility programs) which you can use to simplify the creation of your graphical user interface. The PEG Font Capture utility, PEG Image Converter and PEG Window Builder are each described in the PEG Programming Manual.

### Note

The PEG Programming Manual describes how to port PEG for use with any RTOS and how to configure PEG for your own use. PEG is ready for use with AMX. **No porting** is required. Furthermore, the KwikPeg Configuration Builder is used to configure KwikPeg for use without editing any of the standard PEG header files.

## KwikPeg Reference Manual

The KwikPeg Reference Manual is provided in HTML format suitable for on-line use as you program your GUI application. This manual describes each KwikPeg C++ class, including its derivation, member functions and usage. Programming examples and visual graphic examples are presented for all KwikPeg objects.

### Note

The KwikPeg Reference Manual includes reference material that is NOT provided in either the KwikPeg GUI User's Guide or the PEG Programming Manual.

## Why KwikPeg?

Historically, graphical user interfaces have almost exclusively been in the domain of desktop personal computers. There have been two main factors: the cost of graphical display hardware and the lack of GUI software suitable for use in real-time systems.

In the field of industrial control systems, there have been attempts at graphical presentations, but these have been cumbersome at best and usually expensive. Such systems have typically avoided the use of mainstream video output devices and opted instead for very expensive and functionally limited industrial display terminals.

Today it is common for an embedded system to contain many of the same hardware components found in a desktop computer system. Developers of embedded products have benefited as the sales volume and pricing of desktop computer components have dramatically reduced the cost of graphical display hardware. A wide variety of LCD display panels, VGA display panels, video controller chips and high performance processors capable of driving a graphical interface are now available.

Unfortunately, graphic software has not witnessed the same advances. There have been no graphical interface solutions capable of providing a modern and professional appearance with a small enough footprint for use in embedded systems. Until now!

## What KwikPeg IS

KwikPeg is a GUI generation tool derived from PEG (the Portable Embodied GUI), a professional, high-quality graphic system created by Swell Software, Inc.

KwikPeg is **Portable**. KwikPeg is suitable for use with AMX on any target hardware capable of graphical output. If you have AMX, a C/C++ compiler and hardware capable of pixel addressed graphical output, you can use KwikPeg.

KwikPeg is **Embedded**. For over twenty years, KADAK has been addressing the needs of real-time software developers, long before this overworked term came into vogue. You can expect KwikPeg to meet your embedded products needs.

KwikPeg is **GUI**. KwikPeg provides the building blocks for a powerful and extensible graphical user interface. Using KwikPeg, you can create a graphical presentation with few rivals. The KwikPeg design ensures that it includes all of the advanced GUI features you need today and can accommodate future enhancements. Advanced clipping techniques, font support, graphic image support and smart object methodologies are incorporated into KwikPeg.

## What KwikPeg is NOT

There is some confusion among developers concerning what a GUI generation tool should do and what it should not do. It is therefore helpful to indicate what KwikPeg does not include.

KwikPeg is **not an operating system**. KwikPeg does not provide task switching, memory management, resource management or intertask communication services. Contrary to popular belief, the desktop windowing environment is not part of the operating system (OS). After all, the graphical environment can be significantly updated without improving or otherwise changing the underlying OS.

KwikPeg is **not an application program**. KwikPeg, by itself, will not provide a useful, interactive or informative GUI. You must produce the application which uses KwikPeg to create the windows, dialogs and other objects with which your end user will eventually interact. Of course, it is KwikPeg that makes creating such an application a simple task.

Finally, KwikPeg is **not a PC library**. KwikPeg does include support for PC compatible hardware making KwikPeg ideal for use in PC-like products such as those which incorporate PC/104 embedded PCs. KwikPeg also includes PC support allowing the PC to be used for prototyping your GUI application. However, KwikPeg does not restrict you to PC solutions. Your GUI developed with KwikPeg can be used on any hardware platform which incorporates a display device for which a KwikPeg device driver is available.

## 1.2 General Operation

The KwikPeg GUI and your application operate as illustrated in Figure 1.2-1. All of the components shown in the block diagram are provided with KwikPeg, ready to use with AMX. You simply provide the application.

The KwikPeg GUI sits between your application and the display device. The KwikPeg application interface shields you from any direct involvement with the underlying target hardware, device drivers or operating system.

KwikPeg includes an operating system interface which makes it suitable for use with the AMX real-time operating system. KwikPeg is connected to the RTOS by an OS Interface Module *KP\_OSIF.C*, a C file containing procedures which provide access to AMX services. This module is incorporated into the KwikPeg Library.

The KwikPeg GUI is derived from the KwikPeg Library which is crafted to meet your particular GUI needs and to adapt KwikPeg for use with your specific target hardware. The KwikPeg GUI interacts directly with the KwikPeg device drivers to access a particular video display and optional input devices such as a keyboard and/or mouse. A custom KwikPeg Library is derived for you from a parameter file generated by the KwikPeg Configuration Builder (see Chapter 2).

KwikPeg accesses the display screen through the device driver which handles the hardware interface physically connected to the display device. The KwikPeg device driver interface is described in the PEG Programming Manual. In most instances, the device driver permits direct access by KwikPeg to the video memory, thereby ensuring rapid video updates. Few video device interfaces will benefit from an interrupt driven strategy.

On the other hand, input device drivers for keyboard, mouse or other pointing devices are usually interrupt driven. A separate board driver connects KwikPeg, its device drivers and the OS Interface Module to your target hardware. The board driver module *KP\_BOARD.C* is a C file which is incorporated into the KwikPeg Library.

Finally, your AMX clock driver will generate the fundamental timing needed by KwikPeg.

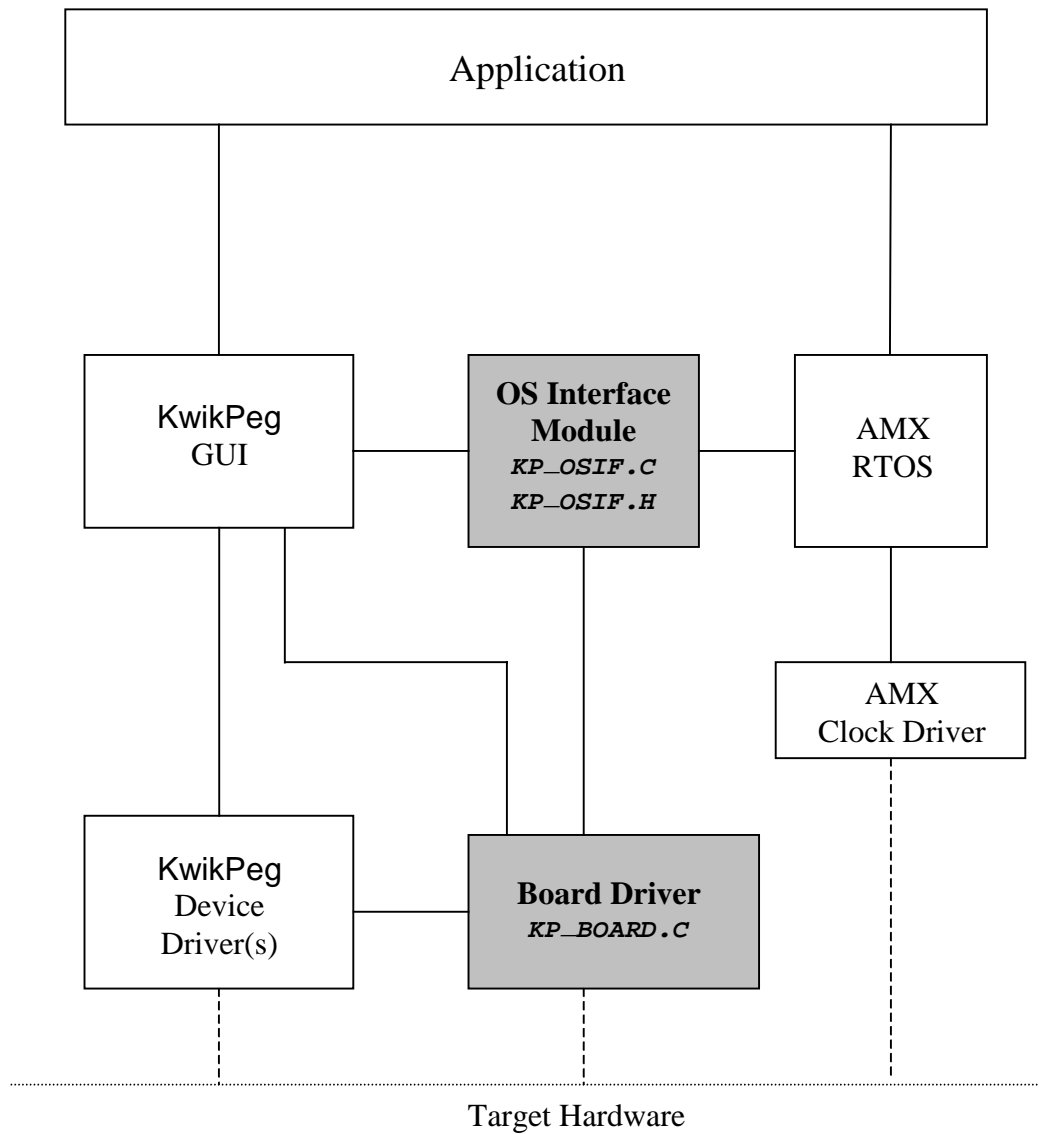


Figure 1.2-1 KwikPeg Application Block Diagram

## KwikPeg Operation

The KwikPeg GUI is ready for use with the AMX Multitasking Kernel. KwikPeg, AMX and your application operate together as illustrated in Figure 1.2-2.

Operation of the KwikPeg GUI is controlled by a single task called the KwikPeg Task. This task begins execution after your application calls procedure `kp_enter()` to start KwikPeg. The KwikPeg Task usually executes at a relatively low priority to avoid interference with other time-critical tasks.

The KwikPeg Task receives timer ticks from AMX through the KwikPeg OS interface. These ticks, KwikPeg's fundamental timing source, occur at the frequency which you specify when you configure your KwikPeg Library.

KwikPeg uses its device drivers to interact with a particular video display and/or input device. The KwikPeg Task and the device drivers cooperate to ensure that video updates and user inputs occur in a timely fashion. Interrupts, if any, generated by the device's hardware interface are serviced by an AMX compatible interrupt service routine (ISR) which calls the device driver's interrupt handler.

Your application tasks interact with KwikPeg using the C/C++ programming interface described in the PEG Programming Manual and the on-line KwikPeg Reference Manual.

At times, KwikPeg may be forced to suspend your application task pending completion of a requested service. How this is done depends on the *magic* of the KwikPeg OS interface. Only the task requesting service is suspended. Other tasks are free to execute and use KwikPeg services. KwikPeg and its OS interface resolve the problems, if any, which may occur when multiple tasks make conflicting demands on the use of the KwikPeg GUI.

Finally, note that most applications will probably include one or more tasks of higher priority than the KwikPeg Task. These tasks, although critical for the success of your application, must not starve the KwikPeg Task's demands for execution time.

### Note

Application tasks which use KwikPeg services can execute at a priority above or below that of the KwikPeg Task.

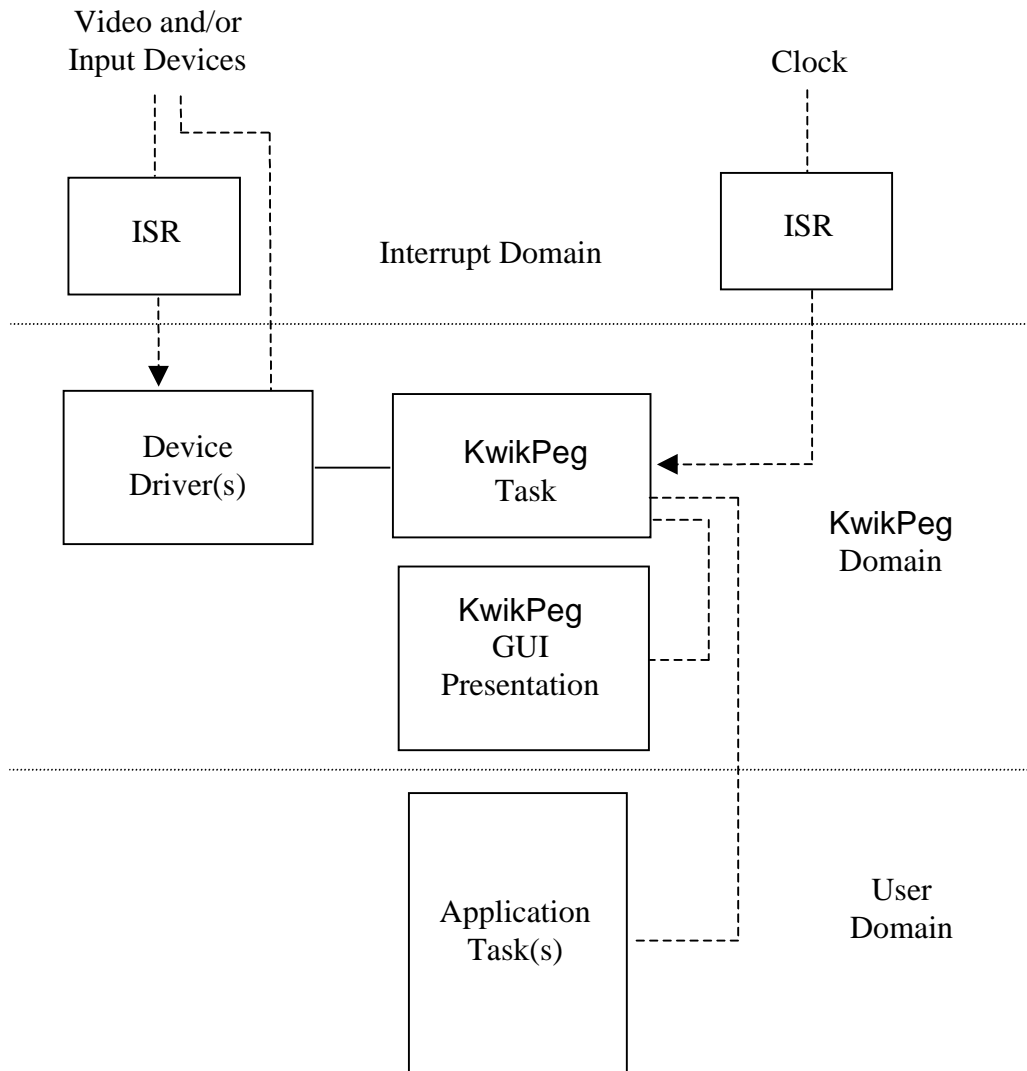


Figure 1.2-2 KwikPeg Operation

## **Byte Ordering and Endianness**

To use a particular video display interface, you must adhere to the byte ordering rules defined by its hardware interface. Doing so is complicated by the fact that not all target processors match the rules adopted by every available video controller.

KwikPeg is independent of the endianness of the processor on which it executes and the video controller to which the processor is connected. All endian dependencies are handled by the device driver and its KwikPeg interface.

## **File System Use**

KwikPeg does not require a file system for its operation. However, if a file system is available, KwikPeg can make use of it. In particular, KwikPeg can access image files from disk for presentation as graphic elements in your GUI. The KwikPeg Universal File System Interface (see Appendix D) can be used to connect KwikPeg to KADAK's AMX/FS™ File System, to the MS-DOS file system or to your own custom file system.

## 1.3 KwikPeg Nomenclature

The following nomenclature standards have been adopted throughout this manual.

Numbers used in this manual are decimal unless otherwise indicated. Hexadecimal numbers are indicated in the format *0xABCD*.

The terminology *A(Table XYZ)* is used to define addresses. It is read as "the address of Table XYZ".

Read/write memory is referred to as RAM. Read only memory (non-volatile storage) is referred to as ROM.

KwikPeg symbol names and reserved words are identified as follows:

<i>kp_pppp</i>	KwikPeg C/C++ procedure name <i>pppp</i>
<i>kpxtttt</i>	KwikPeg structure name of type <i>tttt</i>
<i>xttttyyy</i>	Member <i>yyy</i> of a KwikPeg structure of type <i>tttt</i>
<i>KP_ssssss</i>	Reserved symbols defined in KwikPeg header files
<i>KPT_ssssss</i>	Reserved symbols: basic data type <i>sssss</i>
<i>KPC_ssssss</i>	Reserved symbols: basic color <i>sssss</i>
<i>KP_EReeee</i>	KwikPeg Error Code <i>eeee</i>
<i>KP_WReeee</i>	KwikPeg Warning Code <i>eeee</i>
<i>KP_FEeeee</i>	KwikPeg Fatal Error Code <i>eeee</i>
<i>KPFFFFFF.xxx</i>	KwikPeg reserved filenames
<i>KPnnnFFF.xxx</i>	KwikPeg target and toolset specific filenames
<i>KPZZZFFF.xxx</i>	KwikPeg filenames for application portability
<i>KP_LIB.H</i>	KwikPeg Library Header File

The *nnn* in a KwikPeg filename is the 3-digit KwikPeg part number used by KADAK to identify KwikPeg.

Files with names of the form *KPZZZFFF.xxx* are intended to make KwikPeg less sensitive to the environment in which it is used. For example, the KwikPeg compiler configuration header file *KPZZZCC.H* is used to identify the particular characteristics of the compiler being used to construct your KwikPeg application.

File *KP\_LIB.H* is the KwikPeg Library Header File, an include file which includes the subset of KwikPeg header files needed for compilation of your application C/C++ code. By including file *KP\_LIB.H* in your source modules, your KwikPeg application becomes readily portable to other target processors.

Throughout this manual examples are provided in C and/or C++. Code examples are presented in lower case. File names are shown in upper case. C/C++ code assumes that an *int* is 32 bits on 32-bit processors or 16 bits on 16-bit processors as is common for most C/C++ compilers.

## 1.4 Memory Allocation Requirements

KwikPeg must be able to dynamically allocate and free blocks of memory of varying sizes. KwikPeg supports two memory allocation strategies, both of which are implemented in the OS Interface Module.

The first strategy uses standard C library functions `malloc()` and `free()` to allocate and free memory. The second alternative is to use the memory allocation services provided by the AMX RTOS.

The KwikPeg Library must be configured to select one of the two memory allocation strategies. The strategy is defined by the parameters in your Library Parameter File. The choices are made on the OS property page using the KwikPeg Configuration Builder (see Chapter 2.3).

### Memory Assignment

If you choose to use AMX memory allocation services, then KwikPeg must be configured so that the memory region used by the AMX Memory Manager is provided by your application from one of the following sources:

- (1) a static array in the KwikPeg Library,
- (2) an absolute address in memory or
- (3) a region provided by a memory acquisition procedure `kp_memacquire()`.

### Memory Acquisition Function

If you choose to dynamically assign a memory region to be managed by the AMX Memory Manager for KwikPeg memory allocation, you must provide a memory acquisition function `kp_memacquire()` which KwikPeg will call when it first starts. The prototype for this function is as follows:

```
unsigned long kp_memacquire(char **memp);
```

The memory acquisition function must provide access to a fixed region of  $n$  bytes of memory for use by the AMX Memory Manager. The function must install a pointer to the memory region into the pointer variable referenced as `*memp` and return the value  $n$ , the size of the region.

When KwikPeg shuts down it will call a function `kp_memreturn()` which you must provide to dispose of the memory region, if necessary. The prototype for this function is as follows:

```
int kp_memreturn(char *memp);
```

Parameter `memp` is a pointer to the region of memory previously acquired from your `kp_memacquire()` function. If this function successfully handles the memory region disposal, it must return the value  $0$ . Otherwise, it must return a non-zero value.

## Memory Allocation Protection

When operating in a multitasking environment, the memory allocation services must be thread-safe.

If you choose to use AMX memory management services, the AMX Memory Manager is be used to allocate memory for KwikPeg from a memory pool. These services are inherently thread-safe.

When standard C is used for memory allocation, KwikPeg will enable the memory locking protection in the KwikPeg OS Interface to use an AMX resource semaphore to protect the unsafe services in the C library. In this case, your application must also use the memory allocation services in the KwikPeg OS Interface to ensure thread-safe operation.

## Thread-Safe New and Delete

KwikPeg uses the C++ operators *new* and *delete* to create and delete graphic objects. When operating in the AMX multitasking environment, these operations must be thread-safe. Since the services provided by most C++ libraries are not thread-safe, KwikPeg uses the memory locking mechanism in its OS Interface Module to protect access to the unsafe *new* and *delete* services.

## 1.5 Debugging Aids

KwikPeg includes a number of debug features which, if used effectively, can help you test your GUI application. These features can also be used to provide information to KADAK's technical support staff should you require their assistance.

KwikPeg's debugging services fall into the following categories: GUI testing, breakpoint traps and fatal error detection.

### GUI Testing

Your actual graphical user interface can be built and tested on a Windows workstation using the techniques described in the PEG Programming Manual. This GUI prototyping facility allows you to create, program and test your GUI **before** moving to your target hardware. Your GUI will be presented on the Windows screen with **exactly** the same screen resolution and color depth provided by your target video device driver.

### Breakpoint Traps

KwikPeg can generate a debug trap when it encounters an error condition which is generally not expected in the normal course of events. Such errors are often the result of modifications of private KwikPeg data by errant applications which result in decision conflicts which KwikPeg cannot resolve.

To use KwikPeg's debug trap, you must first build the KwikPeg Libraries to include the extra code necessary to generate the trap. To do so, use the KwikPeg Configuration Builder to edit your KwikPeg Library Parameter File and view the OS property page (see Chapter 2.3). Check the box labeled "Enable debug breakpoint".

Each debug trap generates a call to the KwikPeg breakpoint procedure *kp\_bphit()*. When testing your application, you can place a breakpoint on this procedure to trap all errors detected by KwikPeg.

### Fatal Errors

Some of the errors detected by KwikPeg are serious enough to require that KwikPeg cease operation. To proceed would risk further corruption and would probably lead to a catastrophic collapse in an unpredictable fashion.

When KwikPeg detects such a fatal error, it calls procedure *kp\_panic()* which forces an AMX fatal exit.

When testing, it is always wise to execute with a breakpoint on procedure *kp\_panic()*. You should also have a breakpoint on the AMX fatal exit procedure *cjksfatal* (*ajfatl* and *AAFATL* for AMX 86).

## 1.6 KwikPeg Sample Program - A Tutorial

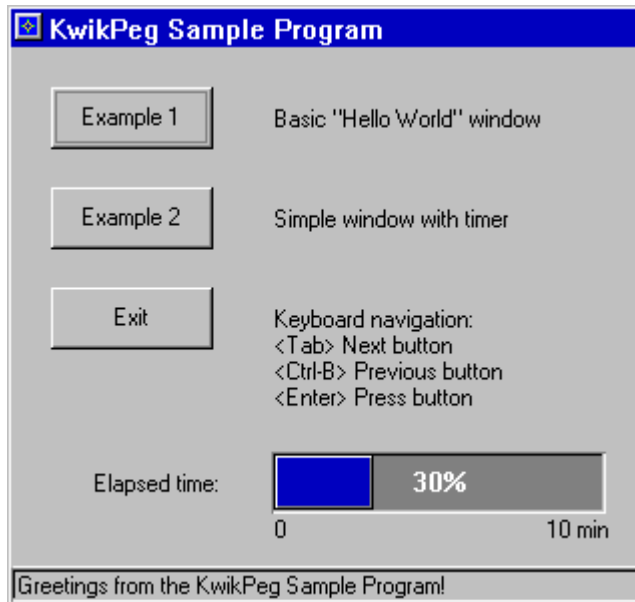
A sample program is provided with KwikPeg to illustrate the use of the KwikPeg GUI within an AMX application.

The sample configuration requires a simple video interface. The sample program is delivered to you ready for use with a particular KwikPeg video device driver. If the driver does not match your video hardware, you will have to create a video device driver for use with your video controller in order to run the sample program.

The sample program also requires a simple input device for user interaction. In most cases, a terminal keyboard connected by a serial link will suffice. You can use the serial UART driver provided with the AMX Sample Program.

The KwikPeg GUI requires a clock for proper timing. The AMX clock driver provided with the AMX Sample Program is suitable for this purpose.

When the KwikPeg Sample Program begins execution with a properly configured video driver, you will see the sample window shown below.



The sample includes two tasks. One task, called the progress task, periodically updates a visual representation of the elapsed execution time. The other task, called the user task, provides the interactive user interface.

The user task creates a window with several buttons with which the user can interact. It also creates the progress bar window. The user task also services the keyboard watching for any of three keys of interest. The Tab and Ctrl-B keys are used to move forward or backward from button to button. The Enter key is used to push a particular button. By pushing the button labeled Exit, the user can terminate the sample program.

The progress task continuously updates the graphical representation of the sample program's elapsed execution time. If the exit button is not pushed, the sample window will automatically close after ten minutes of execution.

## Startup

The KwikPeg Sample Program operates as follows. AMX is launched from the *main()* program. Restart Procedure *rrproc()* starts the user task. A low priority background task is also started to simulate clock interrupts in the absence of a hardware clock.

Once the AMX initialization is complete, the user task finally begins to execute. The user task starts KwikPeg at its entry point *kp\_enter()*. KwikPeg self starts and prepares the KwikPeg Task for execution. In this sample program, the KwikPeg Task has been configured to be of lower priority than the user task. Hence the KwikPeg Task cannot execute until the user task relinquishes control of the processor. The user task therefore calls KwikPeg function *kp\_state()* to wait for the KwikPeg Task to become ready for use.

Once the KwikPeg Task begins to execute, it initializes the video device driver for use by the sample program. It also initializes the serial UART handler for access to the terminal keyboard. It then generates the initial screen image created by the application's *PegAppInitialize()* function.

Once KwikPeg is ready, the user task will resume execution. The user task starts the progress task and then waits for keystrokes from the terminal keyboard. Each key character is encoded for KwikPeg use and passed to the KwikPeg Task for service in the context of the GUI presentation.

In the meantime, the progress task measures the time which has elapsed since the progress task first started. At 200 ms intervals, the progress task signals the KwikPeg Task to update the window which displays elapsed time. The progress task continues to execute periodically until signaled by the GUI's Exit button that a shutdown is imminent.

## Shutdown

When the Exit button in the sample program display window is pushed by the user, the KwikPeg Task closes all windows, performs an orderly shutdown, and ceases to operate. The progress task terminates once it detects that the KwikPeg Task is no longer operational.

The user task also monitors KwikPeg operation. When the user task detects that KwikPeg is no longer running, it stops servicing the keyboard and waits for the KwikPeg Task to end. Finally, the user task requests the operating system to shut down and return to the *main()* function.

## Running the Sample Program

The KwikPeg Sample Program load module is built just like any other KwikPeg application program. The KwikPeg Library Parameter File is first created using the KwikPeg Configuration Builder as described in Chapter 2. Then the KwikPeg Library must be produced. Finally, these KwikPeg modules must be linked with the application, the AMX Library and the C/C++ runtime library to create a load module suitable for testing with a debugger. This construction process is explained in Chapter 3.

Since the sample program cannot operate without a video interface and terminal keyboard, you must provide suitable device drivers for use with your target hardware. Choose the KwikPeg video device driver which matches your video controller or adapt one of them for your own use. Your video device driver will be incorporated into the KwikPeg Library.

You can use the AMX serial UART driver to access the terminal keyboard. You must compile the serial UART driver and link the resulting object module with the sample program (see Chapter 3.6).

With KwikPeg's debug features enabled (see Chapter 1.5), you can place a breakpoint on procedure *kp\_bphit()* to trap all errors detected by KwikPeg. Of course, when you are using AMX, it is always wise to execute with a breakpoint on the AMX fatal exit procedure *cjksfatal* (*ajfat1* or *AAFATL* for AMX 86).

Once you are confident that the KwikPeg Sample Program is operating properly, you may wish to breakpoint your way through the user and progress tasks.

This page left blank intentionally.

## 2. KwikPeg System Configuration

### 2.1 Introduction

Creating an application which uses the KwikPeg Graphical User Interface (GUI) is a two step process. First, the KwikPeg Library must be constructed to reflect the options and features which your application will require and to define the specific characteristics of your target hardware. Then, your application modules must be linked with the KwikPeg Library to create a load module suitable for execution in the target processor.

With many portable GUI tools, this process requires the tedious and error prone task of editing a collection of files with which you have little familiarity. With KwikPeg you simply point and click using the KwikPeg Configuration Builder, a Windows<sup>®</sup> utility which greatly simplifies the process. You still have to pick the correct set of options and define your particular graphical requirements but at least you are concentrating on what you know best, your application.

#### The KwikPeg Library

The KwikPeg Library must be constructed to reflect the options and features which your application will require. This information is kept in a text file called the KwikPeg Library Parameter File which is created and edited for you by the KwikPeg Configuration Builder. This editing process is illustrated in Figure 2.1-1.

From the KwikPeg Library Parameter File, say *GUI\_LIB.UP*, the Builder will generate a make specification file, a text file which can be used to create (make) the KwikPeg Library as described in Chapter 3.2. This file, called the KwikPeg Library Make File *GUI\_LIB.MAK*, is created by merging information from the Library Parameter File with the Library Make Template File *KPnnnLIB.MT*.

The Builder can also generate a copy of the KwikPeg Library Header File *KP\_LIB.H* which is needed to compile KwikPeg source files. This header file will also be included by any application module which uses KwikPeg services. File *KP\_LIB.H* is created by merging information from the Library Parameter File with the Library Header Template File *KPnnnLIB.HT*.

#### Note

If you use the KwikPeg Library Make File *GUI\_LIB.MAK* to create your KwikPeg Library, you do not have to generate the KwikPeg Library Header File *KP\_LIB.H*. It will be created for you during the make process.

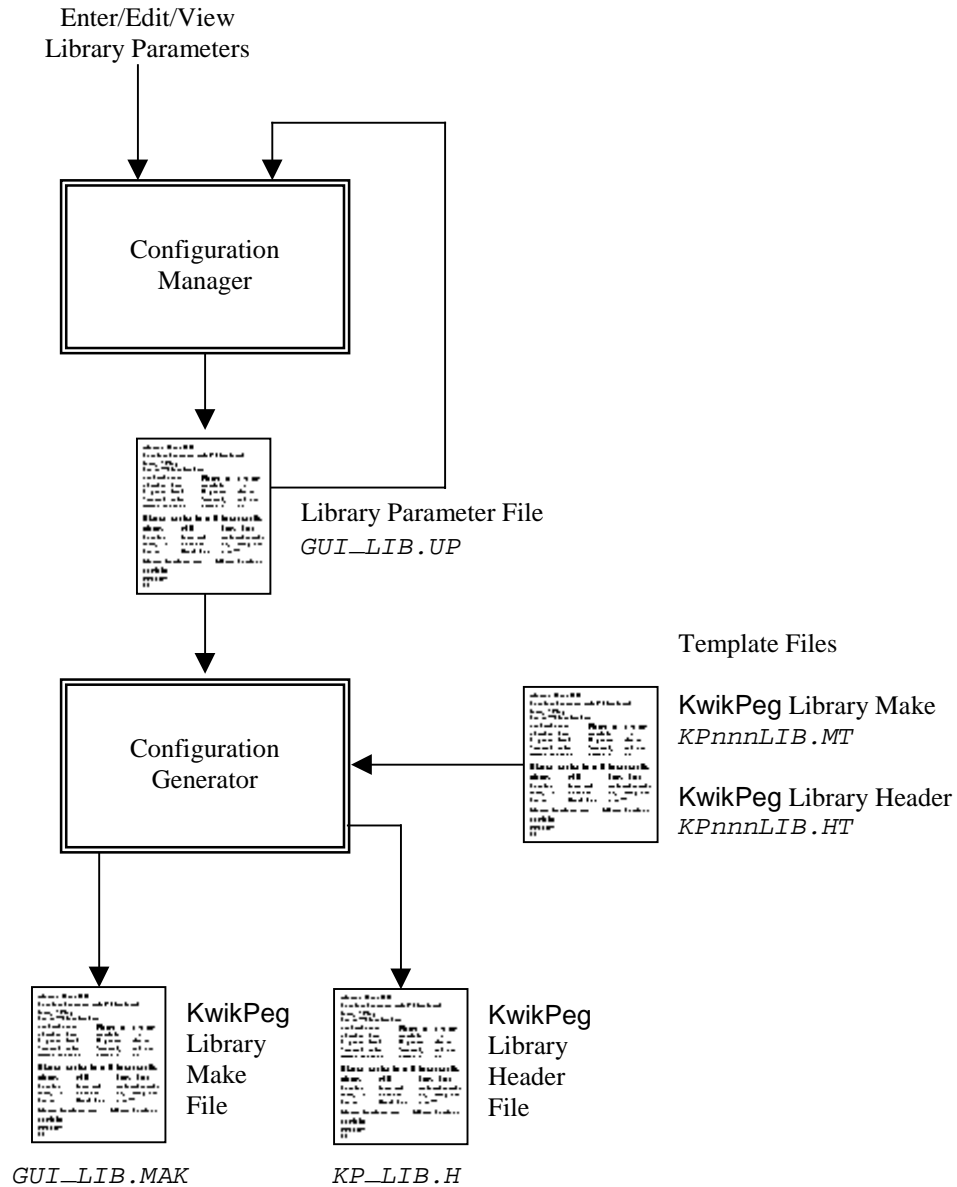


Figure 2.1-1 Creating the KwikPeg Library Make File

## 2.2 KwikPeg Configuration Builder

The KwikPeg Configuration Builder is a software generation tool which can be used to help create your KwikPeg Library. The Builder consists of two components: the Configuration Manager and the Configuration Generator. The Configuration Manager is an interactive utility which allows you to create and edit your Library Parameter File. You can think of the Builder as a very specialized editor.

For convenience, the Configuration Manager has the ability to directly invoke its own copy of the Configuration Generator. The Configuration Generator reads your Library Parameter File and merges the information from it with one of two template files to produce the KwikPeg Library Make File or the KwikPeg Library Header File. This process has been described in Chapter 2.1.

The Configuration Generator is also available as a separate, stand alone DOS utility. This utility program is used by the KwikPeg Library Make File to generate the KwikPeg Library Header File *KP\_LIB.H* with which your application C/C++ files must be compiled. Hence, when you use the KwikPeg Library Make File to create your library, you do not have to generate the KwikPeg Library Header File *KP\_LIB.H* since it will be created for you during the make process.

### Starting the Builder

The KwikPeg Configuration Builder will operate on a PC or compatible running the Microsoft® Windows® operating system.

The KwikPeg Configuration Builder is delivered with the following files.

File	Purpose
<i>KPnnnCM .EXE</i>	KwikPeg Configuration Manager (utility program)
<i>KPnnnCM .CNT</i>	KwikPeg Configuration Manager Help Content File
<i>KPnnnCM .DAT</i>	KwikPeg Configuration Manager Data File
<i>KPnnnCM .HLP</i>	KwikPeg Configuration Manager Help File
<i>KPnnnCG .EXE</i>	KwikPeg Configuration Generator (utility program)
<i>KPnnnLIB.HT</i>	KwikPeg Library Header Template File
<i>KPnnnLIB.MT</i>	KwikPeg Library Make Template File

When KwikPeg is installed on your hard disk, the KwikPeg Configuration Manager utility program and its related files are stored in directory *CFGBLDW* in your KwikPeg installation directory. To start the Configuration Manager, double click on its filename, *KPnnnCM.EXE*. Alternatively, you can create a Windows shortcut to the Manager's filename and then simply double click the shortcut's icon.

## Screen Layout

Figure 2.2-1 illustrates the Configuration Manager's screen layout. The title bar identifies the parameter file being created or edited. Below the title bar is the menu bar from which the operations you wish the Manager to perform can be selected.

Below the menu bar is an optional Toolbar with buttons for many of the most frequently used menu commands. The Toolbar is hidden or made visible using the View Toolbar command on the Edit menu.

The leftmost Toolbar button is used to create a new Library Parameter File.

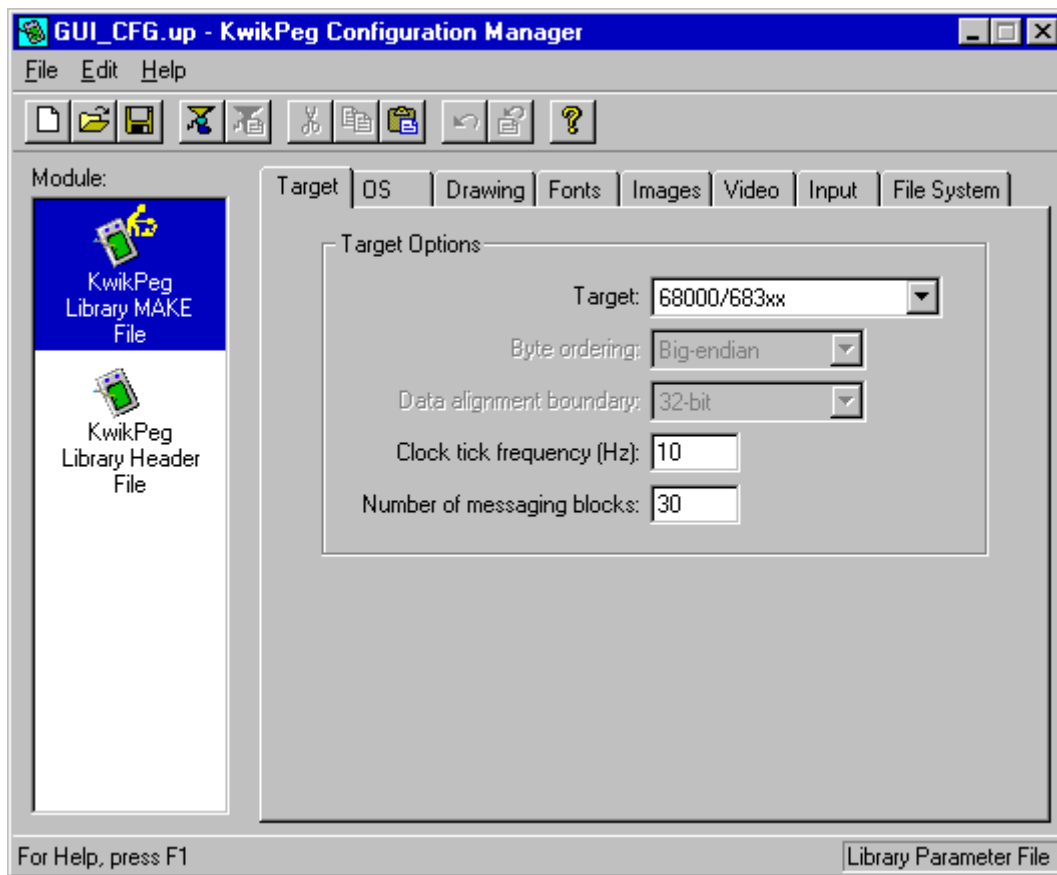


Figure 2.2-1 Configuration Manager Screen Layout

At the bottom of the screen is the status bar. As you select menu items, a brief description of their purpose is displayed in the status bar. If the Configuration Manager encounters an error condition, it presents an error message on the status bar describing the problem and, in many cases, the recommended solution.

Along the left margin of the screen are a set of one or more selector icons. These icons identify the type of output files which the Manager can produce from the parameter file being edited. The example illustrates that when editing a Library Parameter File, the selectors for both the KwikPeg Library MAKE File and the KwikPeg Library Header File are visible.

The center of the screen is used as an interactive viewing window through which you can view and modify your KwikPeg library configuration parameters.

## Menus

All commands to the Configuration Manager are available as items on the menus present on the menu bar. The **File menu** provides the conventional New, Open, Save and Save As... commands for creating and editing your parameter file. It also provides the Exit command.

Once a particular selector icon has been chosen as the currently active selector, the Generate... command on the File menu can be used to generate the corresponding output product. The path to the template file required by the generator to create this product can be defined using the Templates... command on the File menu.

The **Edit menu** provides the conventional Cut, Copy, Paste and Undo editing commands. It also includes an Undo Page command to restore the content of all fields on a property page to undo a series of unwanted edits to the page. The Toolbar is hidden or made visible using the View Toolbar command on the Edit menu.

The **Help menu** provides access to the complete KwikPeg Configuration Manager reference manual. Context sensitive help is also available by pressing the F1 function key or clicking the ? button on the Toolbar.

## Field Editing

When editing a parameter file, a collection of tabbed property sheets is displayed in the central region of the screen. Each tab provides access to a particular property page through which your library configuration parameters can be declared. For instance, if while editing your Library Parameter File, you select the Drawing tab, the Configuration Manager will present a drawing definition window (property page) containing all of the parameters you must provide to completely specify the KwikPeg graphical drawing features you plan to use.

Some fields are boolean options in which all you can do is turn the option on or off by checking or unchecking the associated check box.

Some fields are option fields in which you must select one of a set of options identified with radio buttons or pull down lists. Click on the option button or pick the list item which meets your preference.

Other fields may require numeric or text entry. Parameters are entered or edited in these fields by typing new values or text to replace the current field content. Only displayable characters can be entered. New characters which you enter are inserted at the current cursor position in the field. Right and left arrow, backspace and delete keys may be used to edit the field.

When you are altering a numeric or text field, you tell the Configuration Manager that you are finished editing the field by striking the Enter key. At that point, the Configuration Manager checks the numeric value or text string that you have entered for correctness in the context of the current field. If the value or text string that you have entered is invalid, an error indication is provided on the status bar at the bottom of the screen suggesting how the fault should be corrected.

The Tab and Shift-Tab keys can also be used to complete the editing of a field and move to the next or previous field.

If you have modified some of the fields on a property page and then decide that these modified values are not correct, use the Undo Page command on the Edit menu or Toolbar to force the Configuration Manager to restore the content of all fields on the page to the values which were in effect when you moved to that property page.

When you go to save your parameter file or prepare to move to another property page, the Configuration Manager will validate all parameters on the page which you are leaving. If any parameters are incomplete or inconsistent with each other, you will be forced to fix the problem before being allowed to proceed.

## **Add, Edit and Delete KwikPeg Items**

The KwikPeg Configuration Manager can edit lists of KwikPeg items which are required by your application. The editing techniques used are common to all such lists of items. For example, a property page could be provided to allow you to define one or more KwikPeg widgets for use in your application. In this example, the list of items would be a list of widgets. Note that, although the KwikPeg Configuration Manager is capable of managing a list of widgets, no such widget list is actually presented to you in this implementation.

Pages of this type include a list box at the left side of the property page in which the currently defined items are listed. At the bottom of the list box there may be a counter showing the number of items in the list and the allowable maximum number of such items.

Also below the list are two control buttons labeled Add and Delete. If the allowable maximum number of items is 0 or if all such items have already been defined, the Add button will be disabled. If there are no items defined, the Delete button and all other fields on the page will be disabled.

To add a new item, click on the Add button. A new item with a default identifier will appear at the bottom of the list and will be opened ready for editing. When you enter a valid identifier for the item, your identifier will replace the default in the item list.

To edit an existing item's definition, double click on the item's identifier in the item list. The current values of all of that item's parameters will appear in the property page and the item will be opened ready for editing.

To delete an existing item, click on the item's identifier in the item list. Then click on the Delete button. Be careful because you cannot undo an item deletion.

The items in the item list can be rearranged by dragging an item's identifier to the desired position in the list. You cannot drag an item directly to the end of the list. To do so, first drag the item to precede the last item on the list. Then drag the last item on the list to precede its predecessor on the list.

This page left blank intentionally.

## 2.3 KwikPeg Library Parameter File

When the KwikPeg Library MAKE File or the KwikPeg Library Header File selector icon is the currently active selector, the library configuration's tabbed property sheet is displayed in the central region of the screen. Each tab provides access to a particular property page through which your library configuration parameters can be declared. For instance, if you select the Fonts tab, the Configuration Manager will present a font definition window (property page) containing all of the parameters you can adjust to completely define your proposed font usage.

To create a new Library Parameter File, select New Library Parameter File from the File menu. The Configuration Manager will create a new, as yet unnamed, file using its default KwikPeg library configuration parameters. When you have finished defining or editing your library configuration, select Save As... from the File menu. The Configuration Manager will save your Library Parameter File in the location which you identify using the filename which you provide.

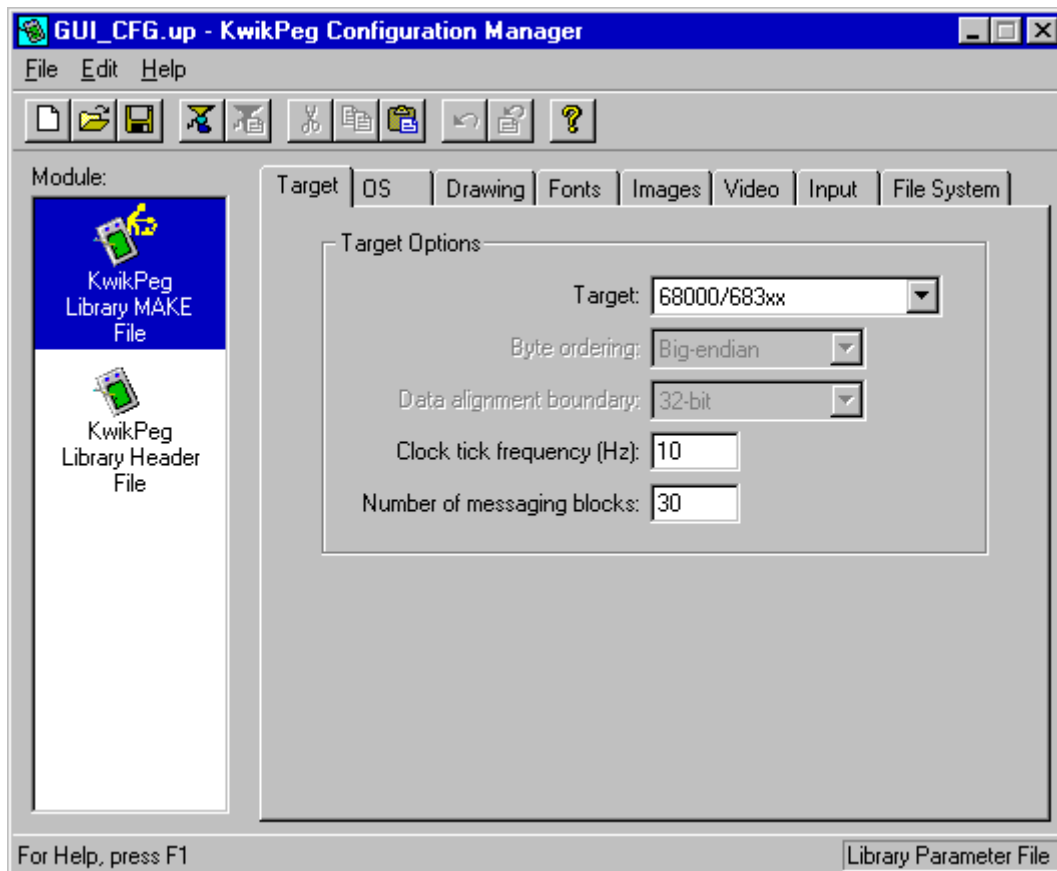
To open an existing Library Parameter File, say *GUI\_LIB.UP*, select Open... from the File menu and enter the file's name and location or browse to find the file. When you have finished defining or editing your library configuration, select Save from the File menu. The Configuration Manager will rename your original Library Parameter File to be *GUI\_LIB.BAK* and create an updated version of the file called *GUI\_LIB.UP*.

When the KwikPeg Library MAKE File selector icon is the currently active selector, the Generate... command on the File menu can be used to generate your Library Make File, say *GUI\_LIB.MAK*. The path to the template file required by the generator to create this product can be defined using the Templates... command on the File menu.

When the KwikPeg Library Header File selector icon is the currently active selector, the Generate... command on the File menu can be used to generate your Library Header File *KP\_LIB.H*. The path to the template file required by the generator to create this product can be defined using the Templates... command on the File menu.

## Target Parameters

The KwikPeg Library must be tailored to operate on a particular target processor. These KwikPeg parameters are edited using the Target property page. The layout of the window is shown below.



## **Target Parameters (continued)**

### **Target Processor**

Select the target processor of interest from those available on the pull down list.

### **Byte Ordering**

From the pull down list, choose the byte ordering scheme (big endian or little endian) used by the target processor's memory system. If the byte ordering method is dictated by the processor, this field will be preset and unalterable.

### **Data Alignment Boundary**

From the pull down list, choose the target processor's natural data alignment (16-bit or 32-bit) for long variables and structures. If the natural data alignment is dictated by the processor, this field will be preset and unalterable.

### **KwikPeg Clock Tick Frequency**

Enter the frequency of the fundamental KwikPeg clock tick. All KwikPeg timing measurements will be based on this frequency.

The KwikPeg clock frequency must be at least 1 Hz. A frequency of 10 Hz or 20 Hz is recommended. Any frequency much above 50 Hz will simply introduce unnecessary execution overhead with little noticeable improvement in display quality or input device response.

Note that KwikPeg must achieve the specified clock frequency using timing services provided by the underlying operating system. You must therefore choose a KwikPeg clock frequency which is derivable by that operating system. Set the KwikPeg clock frequency so that the corresponding period is an integral number of AMX system ticks.

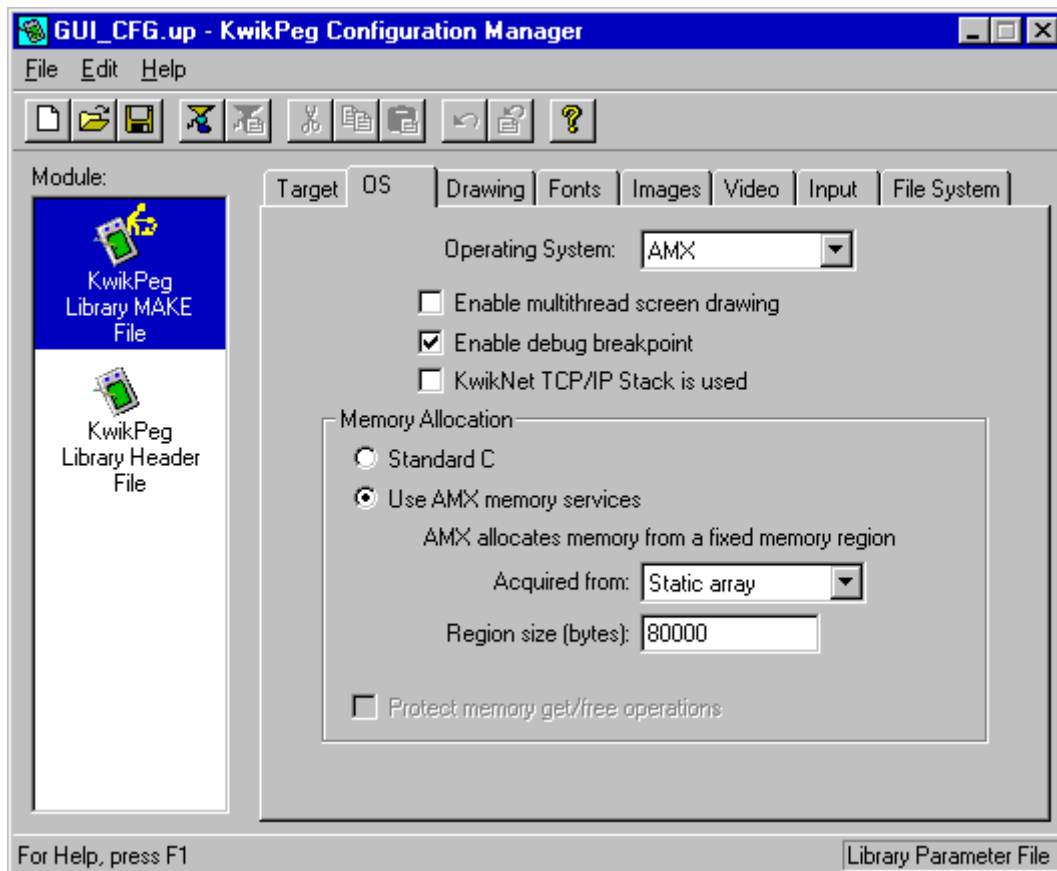
### **Number of Messaging Blocks**

KwikPeg uses messaging blocks for its private internal communication. In most implementations, each of these blocks will require approximately 32 bytes of memory. You may have to increase the number of messaging blocks if any of the following conditions exist.

- You have many tasks generating graphics in separate windows.
- You have many sources of signals which initiate a display update.
- You are using the multithread execution model.

## OS Parameters

The KwikPeg Library must be tailored to operate with a particular operating system. These KwikPeg parameters are edited using the OS property page. The layout of the window is shown below.



## OS Parameters (continued)

### Operating System

From the pull down list, choose the underlying operating system upon which KwikPeg must rely. KwikPeg is ready for use without modification with KADAK's AMX kernels. Choose AMX for any of the 32-bit AMX kernels. Choose AMX 86 for 16-bit, real mode operation on any 80x86/88 processor.

### Enable Multithread Screen Drawing

Check this box if you need to implement one or more tasks which can directly manipulate KwikPeg objects without using the KwikPeg Task to do so. For example, you may have one task which must directly update its own window on the screen without waiting for the KwikPeg Task to first complete all other currently pending screen operations.

When this box is checked, KwikPeg uses its multithread execution model. Enabling this feature can affect performance since KwikPeg must use RTOS semaphore services to avoid potential conflicts between your tasks and the KwikPeg Task. You should avoid using this feature unless essential to your application.

If you do not require multithreaded screen support, leave this box unchecked.

### Enable Debug Breakpoint

Check this box if the KwikPeg Library is to include the code necessary to call the debug breakpoint function *kp\_bphit()* whenever an unexpected runtime error condition is detected. Otherwise leave this box unchecked.

### KwikNet TCP/IP Stack is Used

Check this box if you are using KwikPeg with KADAK's KwikNet TCP/IP Stack for AMX. Otherwise leave this box unchecked.

KwikPeg and KwikNet will provide their own memory allocation services. KwikPeg allocation is separate from KwikNet allocation so that your GUI and the TCP/IP stack can operate independently.

#### Note

A merged form of the KwikPeg and KwikNet OS Interface is available from KADAK to simplify the integration process.

## OS Parameters (continued)

### Memory Allocation

KwikPeg must be able to dynamically allocate and free blocks of memory of varying sizes. KwikPeg uses the memory allocation services in the OS Interface Module *KP\_OSIF.C*. KwikPeg calls OS interface procedure *kp\_osmeminit()* to initialize the memory allocator. Thereafter, KwikPeg calls *kp\_osmemget()* to get a block of memory and *kp\_osmemrls()* to free a previously allocated block.

Two methods of memory allocation are supported.

If you check the Standard C radio button, the OS Interface Module provided with KwikPeg will use standard C library functions *malloc()* and *free()* to allocate and free memory.

If you wish to use the AMX memory allocation services, check the radio button labeled "Use AMX memory services".

### Source and Size of Fixed Memory Region

If the operating system requires a private region of memory for KwikPeg use, you must provide that memory. From the pull down list, choose the method to be used to allocate such a memory region for use by the memory allocator.

If you choose the **Static array** option, you must enter the array size (in bytes) in the Region size field. A character array *kp\_memstore[]* of that size will be declared in the KwikPeg Library. KwikPeg will allocate memory from that array.

If you choose the **malloc()** option, use the Region size field to define the number of bytes to be allocated for KwikPeg use. During KwikPeg's startup initialization, the KwikPeg Task will call C library function *malloc()* once to allocate a memory block of the specified size. KwikPeg will subsequently allocate memory from that single block.

An alternate approach is to choose the **User function** option and provide a function called *kp\_memacquire()* which is prototyped as shown below. The Region size field is unused in this case. When KwikPeg starts up, it will call the function to get the size of your memory region and a pointer to it. The function must install a pointer to a block of *n* bytes of memory into *\*memp* and return the value *n*.

```
unsigned long kp_memacquire(char **memp);
```

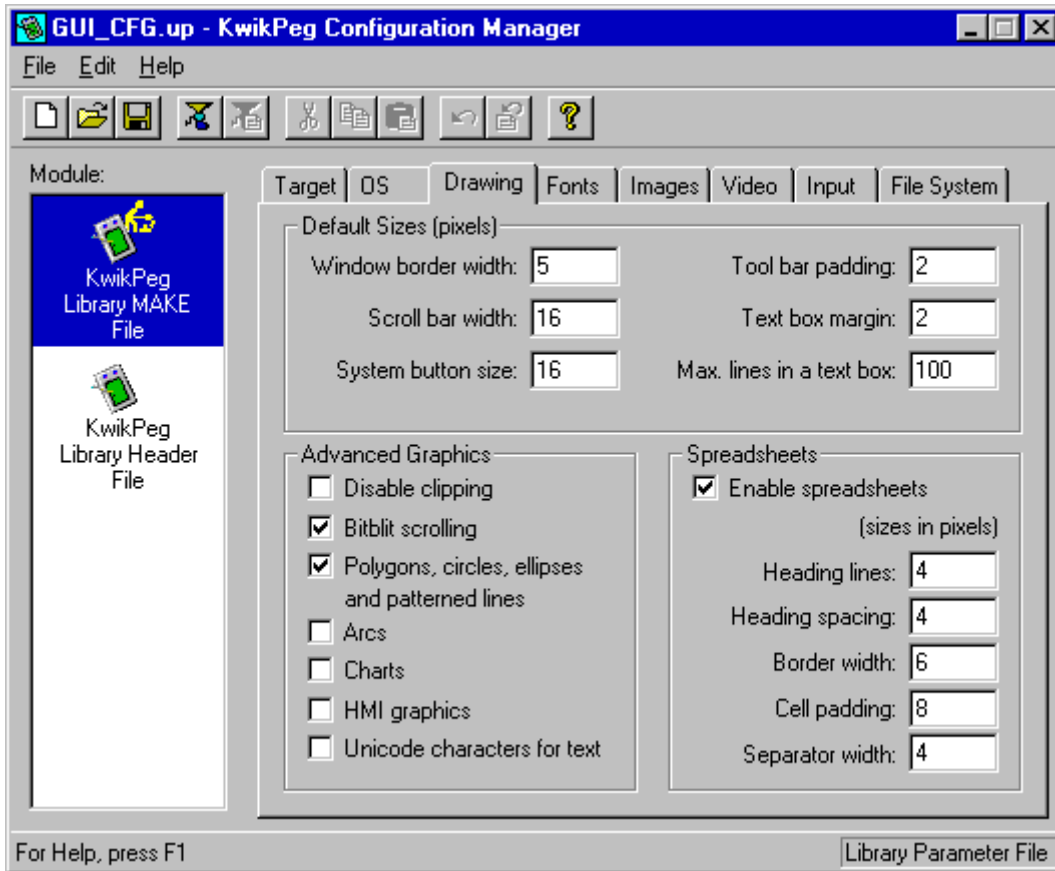
### Estimating Memory Size

Unfortunately, there is no simple formula to estimate the amount of memory which your application will require. For the simplest of GUIs, you will probably need at least 4 Kb of memory. In most cases, you will use less than 16 Kb of memory. However, complex GUIs can require considerably more memory.

Since KwikPeg messaging blocks are allocated dynamically when KwikPeg starts up, be sure to provide enough memory for the total number of messaging blocks which you specified on the Target property page.

## Drawing Parameters

The KwikPeg drawing parameters are edited using the Drawing property page. The layout of the window is shown below.



## **Drawing Parameters (continued)**

### **Window Border Width**

This parameter specifies the default width in pixels of the border KwikPeg adds to windows and controls which it generates. It is the border which adds the illusion of depth or height to an object. The minimum border width is 3 pixels.

### **Scroll Bar Width**

This parameter specifies the default width in pixels of a KwikPeg scrollbar. Unless your application specifies otherwise when it creates a scrollbar, every horizontal or vertical scrollbar generated by KwikPeg will be of this width.

### **System Button Size**

This parameter specifies the default width and height in pixels of the system buttons appearing in a KwikPeg titlebar. Unless your application specifies otherwise when it creates a titlebar, every system button generated on a titlebar will be a square button of this dimension.

### **Tool Bar Padding**

This parameter specifies the number of pixels of padding used to separate the edges of a KwikPeg tool bar panel from the objects placed on the panel.

### **Text Box Margin**

This parameter specifies the number of pixels of padding used to separate the border of a KwikPeg text box from the text placed in the box.

### **Maximum Lines in a Text Box**

This parameter specifies the maximum number of lines of text that can appear in any KwikPeg multiline text box.

## Drawing Parameters (continued)

### Disable Clipping

Check this box if you wish to disable screen clipping. Screen clipping should only be disabled if you have a small screen with few, if any, overlapping windows. When screen clipping is disabled, only the window with the current focus can be updated.

For most applications, leave this box unchecked.

### Bitblit Scrolling

Some KwikPeg drawing objects can be scrolled using one of two different algorithms. In one case, scrolling is performed by repositioning the object's children and redrawing the object. In the second case, a direct video memory move, called a **bitblit** operation, is used to scroll most of the object and only the newly exposed portion of the object is drawn.

If your video controller supports hardware memory move or bitblit operations, check this box so that bitblit scrolling will be used. Using this feature can reduce the scrolling time for a large window from several hundred milliseconds to less than one millisecond.

If you are developing a custom video device driver, leave this box unchecked. Then, if your video controller supports hardware accelerated bitblit operations, add such support to your device driver. Then check this box to let KwikPeg use your driver's bitblit features.

Some of the screen drivers provided with KwikPeg emulate bitblit operations for video devices which do not support the feature. However, there is little improvement in performance over the default scrolling method. In fact, the operations may even be slower for VGA resolutions. Hence, if bitblit operations are not supported by the video hardware, leave this box unchecked.

## **Drawing Parameters (continued)**

### **Polygons, Circles, Ellipses and Patterned Lines**

Check this box if you intend to use KwikPeg's extended graphics capabilities to generate polygons, circles, ellipses, patterned (dashed) lines and related graphic elements. These features do not require floating point support. To omit these drawing services from the KwikPeg Library, leave this box unchecked.

### **Arcs**

Check this box if you intend to use KwikPeg's extended graphics capabilities to generate arcs and related graphic elements. These features do not require floating point support. To omit these drawing services from the KwikPeg Library, leave this box unchecked.

### **Charts**

Check this box if you intend to use KwikPeg's chart generation services. KwikPeg can generate single line charts, multiline charts and stripcharts. To omit these drawing services from the KwikPeg Library, leave this box unchecked.

### **HMI Graphics**

Check this box if you intend to use any of KwikPeg's HMI (Human Machine Interface) classes which simplify the drawing of gauges, dials, scales, storage tanks, valves and other state driven graphic feedback elements. Other useful HMI indicators will be added to the growing KwikPeg repertoire. To omit these drawing services from the KwikPeg Library, leave this box unchecked.

### **Unicode Characters for Text**

Check this box if 16-bit Unicode character encoding will be used by your application for all text strings presented to KwikPeg for graphical rendering. Otherwise, leave this box unchecked.

## **Drawing Parameters (continued)**

### **Spreadsheets**

Check this box if you intend to use KwikPeg's spreadsheet services. To omit these drawing services from the KwikPeg Library, leave this box unchecked.

### **Spreadsheet Heading Lines**

This parameter specifies the maximum number of lines of text which can be used in the column headings of a spreadsheet. You will be able to generate a spreadsheet with zero or more text lines per column heading, up to this maximum. If the value is set to 0, your spreadsheets will be unable to have column headings.

### **Spreadsheet Heading Spacing**

This parameter specifies the extra vertical space in pixels which will be added to the column headings of each spreadsheet generated by KwikPeg. Your application cannot dynamically alter this parameter.

### **Spreadsheet Border Width**

This parameter specifies the width in pixels of the border KwikPeg adds to each spreadsheet which it generates. Your application cannot dynamically alter this parameter.

### **Spreadsheet Cell Padding**

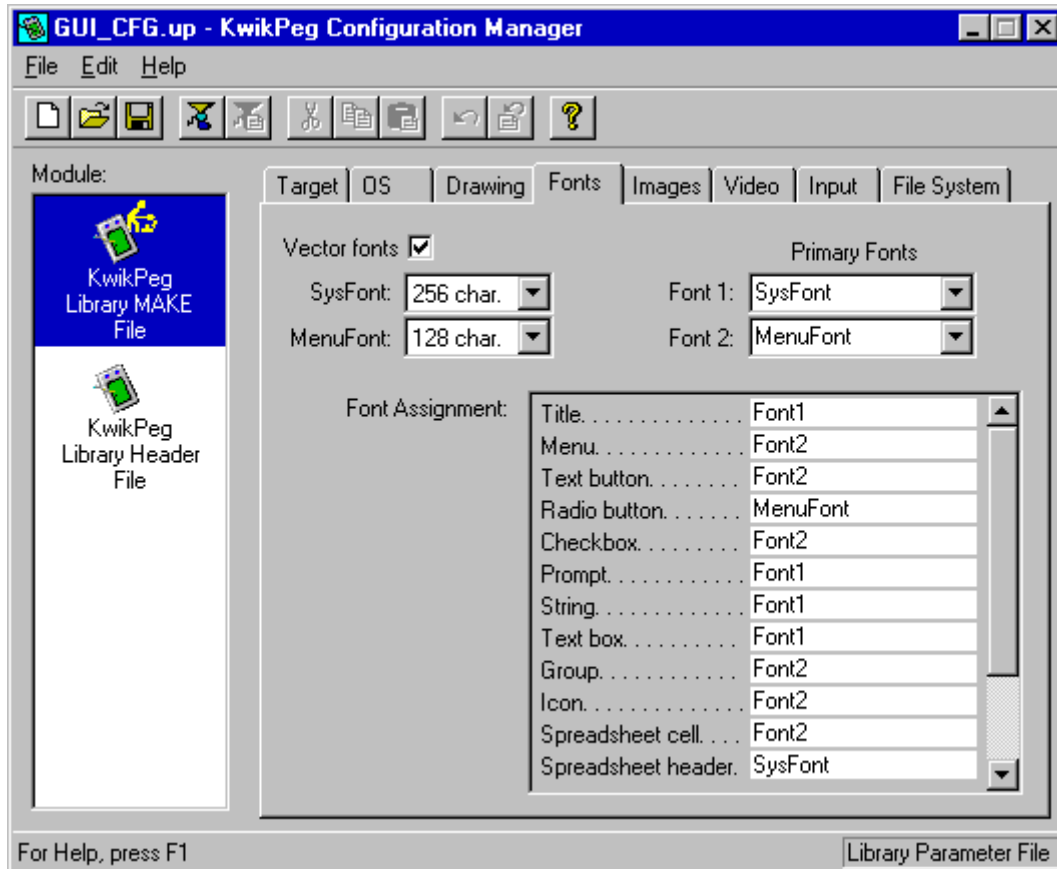
This parameter specifies the default padding in pixels which KwikPeg adds around the content of each spreadsheet cell. When your application creates a spreadsheet, it can adjust the number of pixels used for cell padding.

### **Spreadsheet Separator Width**

This parameter specifies the default width in pixels for the lines, if any, which separate the cells of a spreadsheet. When your application creates a spreadsheet, it can adjust the width of the cell separator lines.

## Font Parameters

The KwikPeg font usage parameters are edited using the Fonts property page. The layout of the window is shown below.



## Vector Fonts

Check this box if you intend to use the scalable KwikPeg vector fonts. If you use vector fonts, your application can create a font with a specific point size and attributes at runtime. You can then use your dynamically created font for text generation. When the customized font is no longer required, it can be deleted. To omit support for vector fonts from the KwikPeg Library, leave this box unchecked.

## SysFont and MenuFont Character Sets

KwikPeg provides two fonts for your use: one named *SysFont* and a smaller variant named *MenuFont*. Each font is available in a full 256 character set. To reduce memory usage, you can restrict each font set to the first 128 characters of the set. Choose the size of the character set for each of these fonts from the pull down list. To eliminate the font in its entirety, select option None from the pull down list.

## Font Parameters (continued)

### Primary Fonts 1 and 2

KwikPeg defines two primary fonts identified as Font1 and Font2. You are free to assign any available font to each of the primary fonts. From the pull down list, choose the KwikPeg SysFont or MenuFont for use as Font1 or Font2. Alternatively, provide the name of one of your own custom fonts by typing its name in place of the currently displayed name. For example, if Font2 is currently MenuFont, edit the string MenuFont to be MyArial\_8 to select a custom 8 point Arial font known by that name. To assign a minimal (empty) font, use font name NullFont. Font1 and Font2 can be referenced by your application software using font IDs `PEG_FONT1_FONT` and `PEG_FONT2_FONT` respectively.

### Font Assignments

KwikPeg defines the particular font which is to be used by default for each graphic object requiring text decoration. The objects requiring fonts are divided into the categories listed on the Fonts property page. For each category you can choose the font to be used when an object of that category is created.

To change the font for a particular category of object, click on that object's current font. From the pull down list, choose one of the four available fonts. You will see that the KwikPeg SysFont or MenuFont fonts are always available. However, objects can also be assigned either of the primary fonts Font1 or Font2. The object will be decorated using text in the actual font currently assigned to primary font Font1 (or Font2). Note that any font change made to a primary font automatically ripples to all of the objects which use that primary font.

You can also provide the name of one of your own custom fonts by typing its name in place of the currently displayed font name. For example, if Font1 is currently being used for Prompt text, edit the string Font1 to be MyTNR\_Italic\_10 to select a custom 10 point Times New Roman italic font known by that name. To assign a minimal (empty) font, use font name NullFont.

### Custom Fonts

Custom fonts can be created using the PEG Font Capture tool, the Windows<sup>®</sup> utility described in the PEG<sup>™</sup> Programming Manual. Note that capturing a font does not grant you license to use that font.

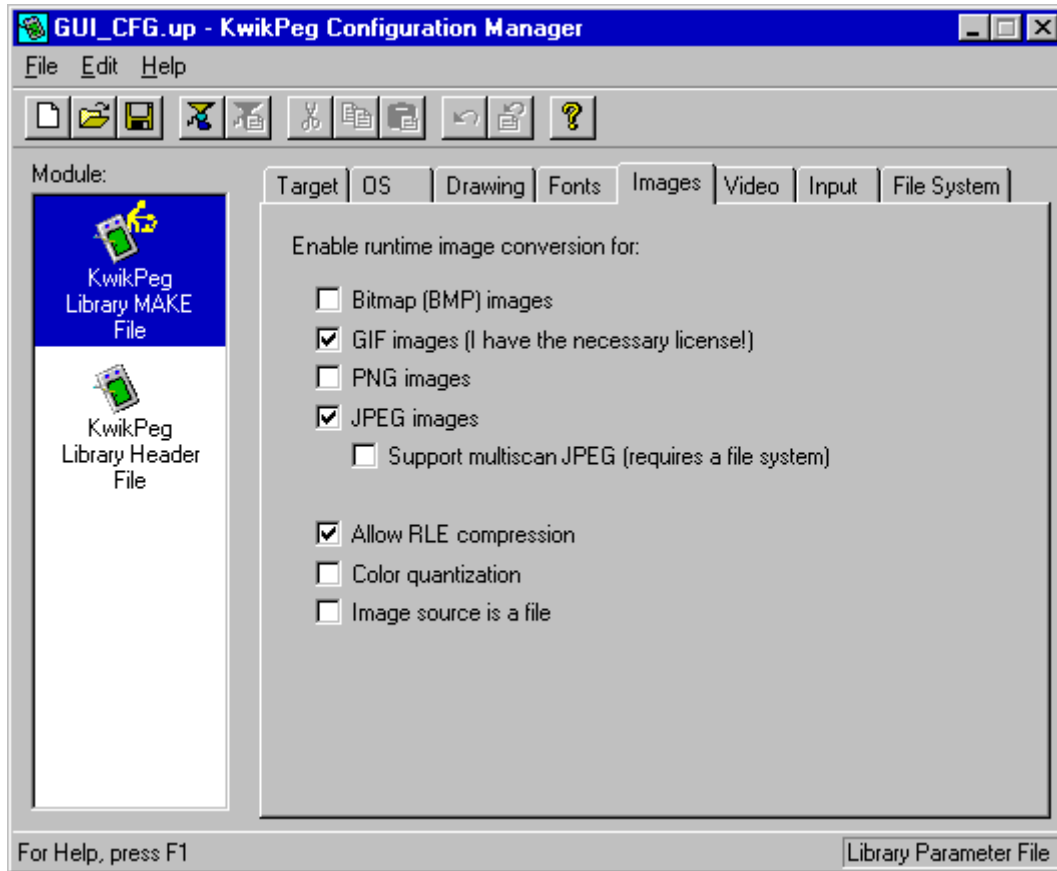
A custom font can also be created at runtime as long as you ensure that such a font is contained in a public instance of a KwikPeg *PegFont* structure that is initialized before any reference to the font is made.

**\*\* WARNING \*\***

It is your responsibility to obtain the necessary license, if any, required to use any font other than those provided with KwikPeg.

## Image Conversion Parameters

The KwikPeg image conversion parameters are edited using the Images property page. The layout of the window is shown below.



### Enable Runtime Image Conversion

Check a particular box if you intend to dynamically convert bitmap (*BMP*), *GIF*, *PNG* or *JPEG* images to the internal bitmap format used by KwikPeg for image rendering. To omit support for a particular type of runtime image conversion from the KwikPeg Library, leave the box unchecked.

### Support Multiscan JPEG

If you intend to convert multiscan *JPEG* images, check this box. To convert such images, KwikPeg requires a file system for temporary storage. If multiscan *JPEG* support is not required, leave the box unchecked.

## Image Conversion Parameters (continued)

### Allow RLE Compression

Check this box if you wish the converted images to be compressed using run length encoding (RLE) techniques. Most images will benefit from RLE compression, resulting in smaller bitmap images in memory. KwikPeg will not compress an image if the result is not a smaller bitmap image in memory. To omit support for RLE compression from the KwikPeg Library, leave this box unchecked.

### Color Quantization

Check this box if color quantization techniques are to be used to derive an optimal palette for the converted images. To omit support for runtime color quantization from the KwikPeg Library, leave this box unchecked. Most converted images will not require color quantization.

### Image Source is a File

Check this box if the images to be converted are to be read from files. Obviously, your application will require a file system in this case. If you leave this box unchecked, each image to be converted must be presented to KwikPeg using a data streaming callback function which you must provide.

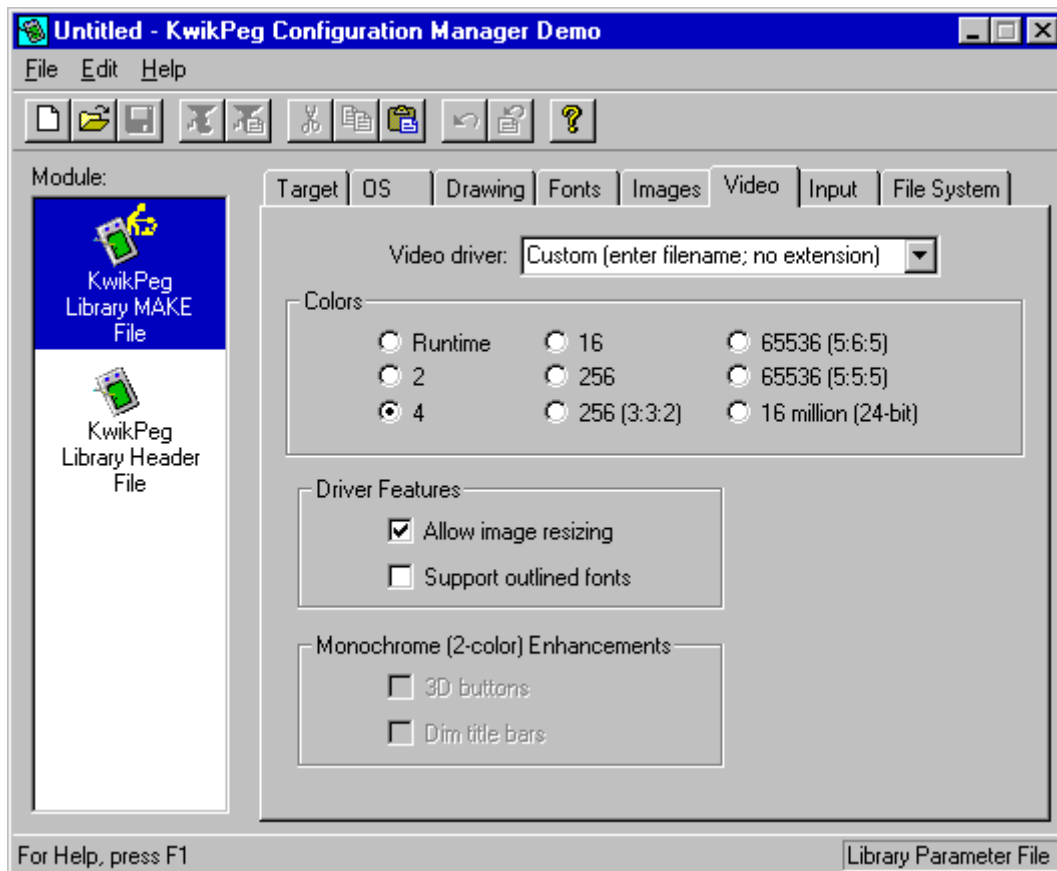
#### **\*\* WARNING \*\***

KwikPeg uses the LZW decompression algorithm to read and decode *GIF* images. The LZW algorithm is patented; the patent is owned by Unisys Corporation. Usage of LZW requires that you obtain a usage license directly from Unisys. Software providers such as KADAK Products Ltd. and Swell Software, Inc. are not allowed to provide a transferable usage license. It is therefore your responsibility to obtain an LZW usage license if your application will perform runtime conversion of *GIF* images.

By providing you with a *GIF* decompression capability, KADAK and Swell Software in no way imply that an LZW license has been obtained and/or granted to you. Nor does either assume responsibility for the actions of any user who willfully violates the Unisys LZW patent by using the KwikPeg *GIF* decompression services without first obtaining an LZW license from Unisys.

## Video Parameters

KwikPeg includes a number of video device drivers. The video device driver is selected and its parameters edited using the Video property page. The layout of the window is shown below.



## **Video Parameters (continued)**

### **Video Driver**

From the pull down list, choose one of the available KwikPeg video device drivers. These device drivers are defined in the KwikPeg Configuration Manager Data File *KPnnnCM.DAT* which is located in the KwikPeg *KPGnnn\CFGBLDW* installation directory. Alternatively, pick the Custom device driver and edit the string in the list box, replacing it with the filename of your video device driver. Enter the filename with no file extension. The driver header and source files must have file extensions *.HPP* and *.CPP* respectively.

Once a video device driver is selected, any options on the Video property page which are not supported by the video device driver will be dimmed.

### **Number of Colors**

Choose the number of colors which your KwikPeg Library must support. If the video device driver supports more than one color option, choose the minimum number of colors which meet the image generation requirements of your application.

Some video device drivers allow the number of colors to be dynamically determined at runtime. If the video device driver supports this feature, you can choose the Runtime option to allow the video device driver to specify the color count for KwikPeg.

### **Allow Image Resizing**

Check this box if you wish to be able to resize bitmap images. Your video device driver must provide the necessary support. All of the video device drivers and template drivers provided with KwikPeg support image resizing. To omit support for runtime image resizing from the KwikPeg Library, leave this box unchecked.

### **Support Outlined Fonts**

Check this box if you wish to be able to draw outlined fonts captured by the PEG Font Capture utility. Your video device driver must provide the necessary support. All of the video device drivers and template drivers provided with KwikPeg support outlined fonts. To omit support for outlined fonts from the KwikPeg Library, leave this box unchecked.

### **3D Monochrome Buttons**

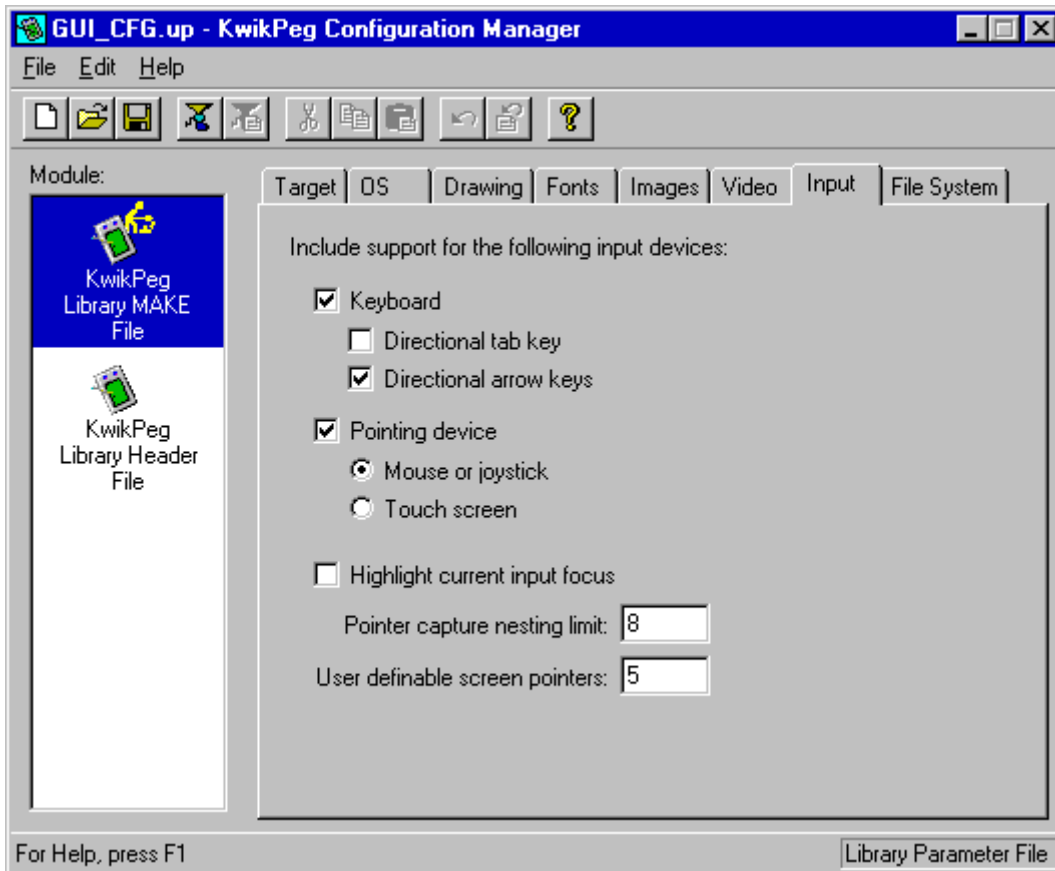
Check this box if you want buttons to be rendered as 3-dimensional images on monochrome (2-color) displays. Otherwise, leave this box unchecked.

### **Dim Monochrome Title Bars**

Check this box if you want title bars to be dimmed on monochrome (2-color) displays so that active and inactive windows will be distinguishable. Otherwise, leave this box unchecked.

## Input Device Parameters

KwikPeg includes support for a number of input and pointing devices. The input and/or point device is selected and its parameters are edited using the Input property page. The layout of the window is shown below.



## **Input Device Parameters (continued)**

### **Keyboard**

Check this box if you require keyboard support for navigation through menus, windows and dialogs. Keyboard support may range from a full QWERTY keyboard to a small set of membrane keys. Full navigation and operation can be accomplished with as few as three unique input keys. To omit keyboard support from the KwikPeg Library, leave this box unchecked.

### **Directional Tab Key**

Check this box if you want to navigate through your windows in a "next/previous" fashion using the tab key.

### **Directional Arrow Keys**

Check this box if you want to navigate through your windows in an "up/down/left/right" fashion using the arrow keys.

### **Pointing Device**

Check this box if your application requires support for a pointing device such as a mouse, joystick or touch screen. Otherwise, leave this box unchecked.

### **Mouse or Joystick**

To use a mouse or joystick, check this radio button. You must provide a mouse or joystick device driver for use with your application. The examples provided with KwikPeg can be adapted to meet your needs.

### **Touch Screen**

To include support for a touch screen, check this radio button. You must provide a touch screen device driver for use with your application. The examples provided with KwikPeg can be tailored to your needs.

## **Input Device Parameters (continued)**

### **Highlight Current Input Focus**

Check this box if you want KwikPeg to highlight the object on the screen which has the current input focus. This feature is particularly useful with a keyboard in order to provide visible feedback when using keystrokes for navigation. If you do not want KwikPeg to add this extra bit of input focus visibility, leave this box unchecked.

### **Pointer Capture Nesting Limit**

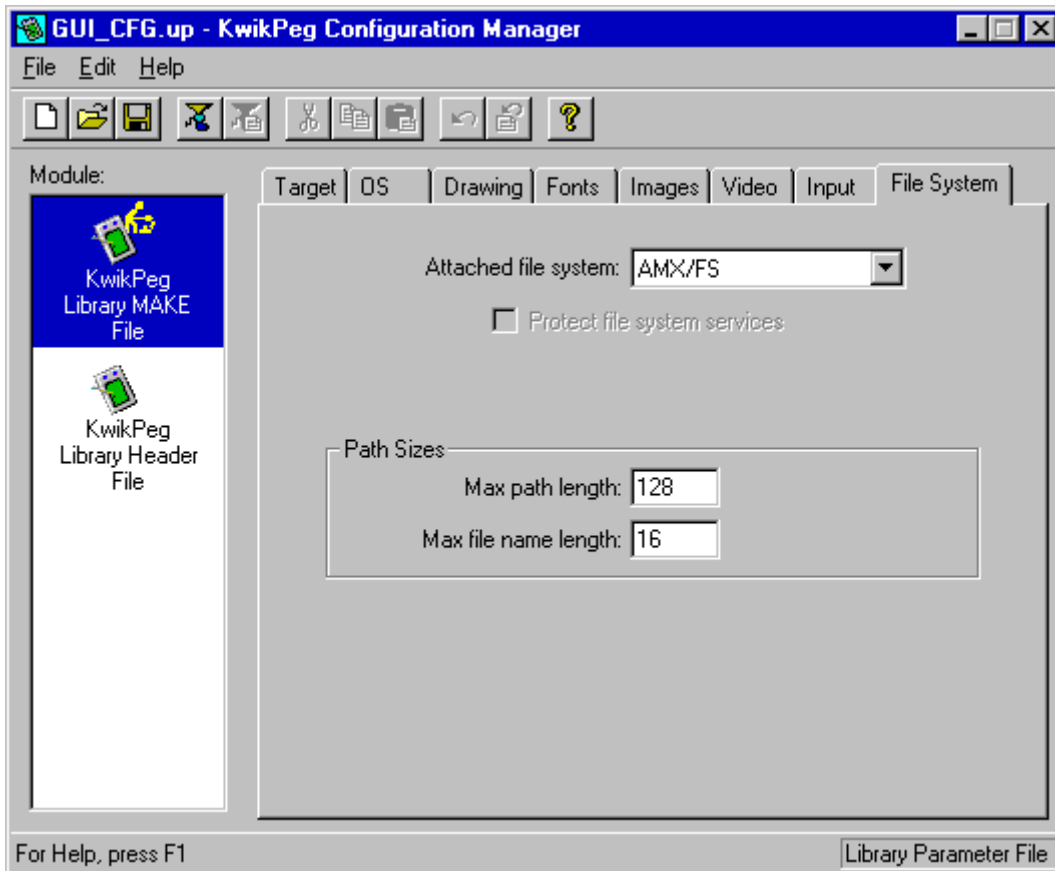
KwikPeg allows any drawing object to capture the input pointer. Subsequent input operations are then sent directly to that object. To accommodate the recursive use of objects which capture the input pointer, KwikPeg allows pointer capture requests to be nested. This parameter defines the maximum number of times the pointer can be captured without being released. If your GUI attempts to capture the pointer while nested to this limit, the capture request will fail. You can use the KwikPeg debug breakpoint trap to detect such an occurrence.

### **User Definable Screen Pointers**

In addition to the standard system pointers (cursors), KwikPeg allows you to create custom pointers via the *PegScreen::AddPointerType()* method. This parameter defines the maximum number of custom pointers that can be added.

## File System Parameters

The KwikPeg Universal File System (UFS) interface forms part of the KwikPeg Library. The Universal File System parameters are edited using the File System property page. The layout of the window is shown below.



## **Universal File System Parameters (continued)**

### **Attached File System**

From the pull down list, choose the type of underlying file system to which the Universal File System must connect. If you are not using any KwikPeg features which require a file system, select option None.

If you are using KwikPeg with the KwikNet TCP/IP Stack, you can use the KwikNet Universal File System if it has been included as part of your KwikNet configuration. To do so, select option KwikNet UFS from the pull down list.

### **Protect File System Services**

When operating in a multitasking environment, the file system services must be thread-safe. If the file system you have chosen to use is safe, leave this box unchecked. Otherwise, check this box and KwikPeg will use its file system locking mechanism to protect access to the unsafe file system services. In a single threaded environment, file system services are inherently thread-safe. Hence, leave this box unchecked.

### **Maximum Path and File Name Lengths**

Specify the maximum number of characters which can appear in a path name string used to reference a directory location. The path length must include room for a terminating '\0' character. The path length excludes any file name.

Specify the maximum number of characters which can appear in a file name string used to identify any file. The file name includes the base name and extension(s) and any separating characters. The file name length must include room for a terminating '\0' character. The file name length excludes any path information.

There is no reason to set the path or file name lengths any greater than the maximum allowed by the selected file system. For AMX/FS and MS-DOS file systems, a path length of 128 and a file name length of 16 will be adequate.

To minimize memory waste in embedded applications in which short paths are the norm, the maximum path length can be reduced.

## 3. KwikPeg System Construction

### 3.1 Building an Application

KwikPeg is ready for use with the AMX kernel. Hence, you can immediately construct a KwikPeg Library which meets your particular needs and build an actual KwikPeg application. The sample program provided with KwikPeg is an example which you can use either for guidance or as a starting point for your own application.

To build a KwikPeg application you must perform the following steps.

1. Using the KwikPeg Configuration Builder, create and/or edit a Library Parameter File to select the KwikPeg features which your application requires. On the OS property page, enable KwikPeg's debug breakpoint feature to assist you during initial testing. Use the builder to generate your KwikPeg Library Make File.
2. If none of the available KwikPeg video device drivers meet your needs, create a custom device driver using the most suitable driver as a starting point. If your application must support an input device, adapt one of the available input device drivers for use with your hardware.
3. If necessary, adapt the KwikPeg board driver *KP\_BOARD.C* to accommodate your target processor, device interfaces and interrupt management scheme.
4. Using the KwikPeg Library Make File generated in step 1, create your KwikPeg Library following the procedure to be described in Chapter 3.2.
5. Finally, create a make file which your make utility can use to build your application. It must compile your application modules and your KwikPeg video and input device drivers. It can then link the resulting object modules with your KwikPeg Library, the AMX Library and your C/C++ run-time library to create an executable load module. Follow the compilation and linking recommendations presented in Chapters 3.3 and 3.4.
6. Use your software debugger and/or in-circuit emulator tools to transfer your load module to your target hardware. When testing, you should execute your application with a breakpoint on KwikPeg procedure *kp\_bphit()* so that you can readily detect fatal configuration or programming errors or unusual operation of the KwikPeg GUI.

## 3.2 Making the KwikPeg Library

To build the KwikPeg Library, you will need a make utility capable of running your C/C++ compiler and object librarian (archiver). The library construction process is illustrated in Figure 3.2-1. The shaded blocks indicate modules which adapt the make process to accommodate your software development tools. Since you are using KwikPeg with AMX, these modules are ready for use without modification.

Your custom KwikPeg Library is created from the KwikPeg Library Parameter File, a text file describing the GUI features and options which your application requires. This file is created and edited using the KwikPeg Configuration Builder as described in Chapter 2.

The KwikPeg Configuration Builder uses the information in your Library Parameter File to generate a KwikPeg Library Make File. This make file is suitable for use with Microsoft's *NMAKE* utility. The make file purposely avoids constructs and directives that tend to vary among make utilities. Hence, you should have little difficulty using this make file with your own make utility if you so choose.

The make utility uses your C/C++ compiler and object librarian to generate the KwikPeg Library from the KwikPeg source modules.

All KwikPeg C/C++ source files include a **compiler configuration header file** *KPZZZCC.H*. This file identifies the characteristics of your C/C++ compiler. This file is also used to optimize code sequences within KwikPeg modules by taking advantage of compiler specific features such as in-line code, assembly language functions and C/C++ library macros or functions. A number of variants of this module are provided with KwikPeg ready for use with popular compilers on a variety of target processors.

As you would probably expect, the make file does not know how to run your C/C++ compiler and object librarian. This information is provided in a file called *KPZZZCC.INC* which the make process automatically includes. This file, called a **tailoring file**, is used to tailor the library construction process to accommodate your make utility's syntax for implicit rules. It also provides the command sequences necessary to invoke your C/C++ compiler and object librarian. KwikPeg is shipped with a number of tailoring files ready for use with Microsoft's *NMAKE* utility and many popular compilers.

Most object librarians allow you to provide a list of the files which form a library in a file which KADAK calls a **library specification file**. KwikPeg is shipped with library specification files for each of the toolsets with which the KwikPeg Library has been built.

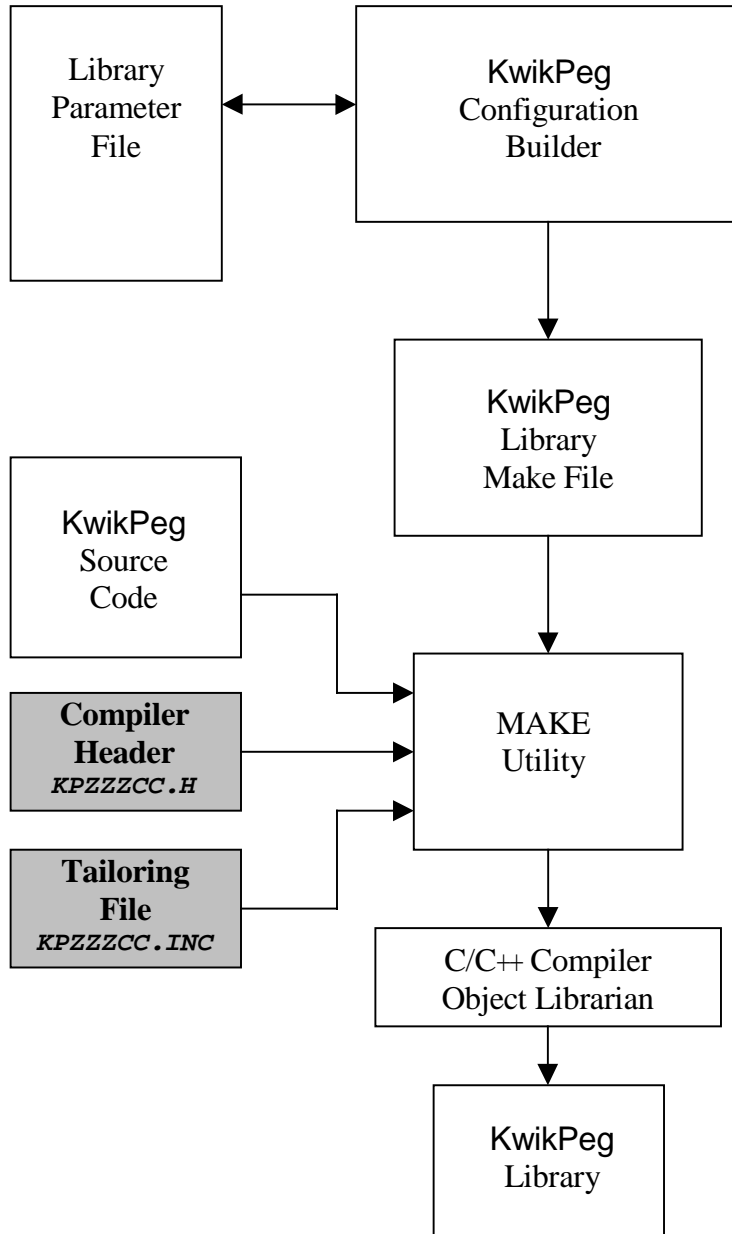


Figure 3.2-1 KwikPeg Library Construction

## KwikPeg Directories and Files

In order to use KwikPeg, you must have installed the standard PEG distribution from Swell Software, Inc. Once PEG has been installed in directory *PEG*, the source code for the KwikPeg Library will reside in the following subdirectories.

<i>PEG\INCLUDE</i>	Library header files
<i>PEG\INCLUDE\SCRNDRV</i>	Video driver header files
<i>PEG\SOURCE</i>	Library source code
<i>PEG\SOURCE\SCRNDRV</i>	Video driver source code

The make process depends upon the structure of the KwikPeg installation directory *KPGnnn*. When KwikPeg is installed, the following subdirectories are created within directory *KPGnnn*.

<i>CFGBLDW</i>	KwikPeg Configuration Builder; template files
<i>ERR</i>	Construction error summary
<i>LIB</i>	KwikPeg Library will be built here
<i>MAKE</i>	KwikPeg make directory
<i>PEGFILES</i>	PEG integration header and source files
<i>TOOLUU</i>	Toolset specific files must be installed here
<i>TOOLUU\CC_H</i>	Toolset specific compiler configuration header files
<i>TOOLUU\CC_LBM</i>	Toolset specific library specification files
<i>TOOLUU\TF_MSOFT</i>	Toolset specific tailoring files for Microsoft <i>NMAKE</i>

Directory *PEGFILES* contains integration header and source files that are copied to your PEG installation directory when KwikPeg is installed. Other directories containing the sample program and GUI examples will also be present but are not involved in the library construction process.

## Compiler Configuration Header File

All KwikPeg C/C++ source files include a **compiler configuration header file** *KPZZZCC.H*. This file identifies the characteristics of your C/C++ compiler. This file is also used to optimize code sequences within KwikPeg modules by taking advantage of compiler specific features such as in-line code, assembly language functions and C/C++ library macros or functions. All compiler configuration header files are located in directory *TOOLUU\CC\_H* in the KwikPeg installation directory *KPGnnn*.

KADAK uses a 2 or 3 character toolset mnemonic *xxx* to identify each supported toolset combination. For toolset *xxx*, the header file is named *H\_tttxxx.vvv*. The mnemonic *ttt* identifies the target processor. The extension *vvv* identifies the first version of the compiler for toolset *xxx* with which the header file was tested. The header file can be used with subsequent versions of the tools until some change in their method of operation requires a header file update. For example, file *H\_PPCDA.41* was first used to create the PowerPC KwikPeg Library using the Diab-SDS (toolset *DA*) C/C++ compiler. In this case, although the file identifies the toolset version as *41*, the file is known to be valid for Diab-SDS v4.1, v4.2 and v4.3.

The KwikPeg compiler configuration header files *H\_tttXXX.vvv* listed in Figure 3.2-2 have been created and tested for use with KwikPeg and the AMX kernel. The files will be found in installation directory *TOOLUU\CC\_H*. The KwikPeg product is always delivered to you with the most recent available set of these header files.

When using one of these files, you must copy the file and rename it *KPZZZCC.H*.

<b>File</b>	<b>Target</b>	<b>Compiler</b>
<i>H__86MC.15</i>	80x86 (real mode)	Microsoft 16-bit Visual C/C++ v1.5, v1.52
<i>H__86PD.50</i>	80x86 (real mode)	Paradigm 16-bit C/C++ v5.0, v6.0
<i>H__86TC.50</i>	80x86 (real mode)	Borland 16-bit C/C++ v5.0
<i>H__86WC.110</i>	80x86 (real mode)	WATCOM (Sybase) 16-bit C/C++ v11.0
<i>H_386PD.60</i>	80x86 (protected mode)	Paradigm 32-bit C/C++ v6.0
<i>H_68KDA.42</i>	68000	Diab-SDS C/C++ v4.2 and up
<i>H_68KIM.92</i>	68000	TASKING (Intermetrics) C/C++ v9.2r0
<i>H_68KME.20</i>	68000	Metrowerks C/C++ v2.0 and up
<i>H_68KMR.45</i>	68000	Mentor Graphics (Microtec) C/C++ v4.5G
<i>H_CFDA.42</i>	ColdFire	Diab-SDS C/C++ v4.2 and up
<i>H_CFME.25</i>	ColdFire	Metrowerks C/C++ v2.5
<i>H_PPCDA.41</i>	PowerPC	Diab-SDS C/C++ v4.1 and up
<i>H_PPCME.42</i>	PowerPC	Metrowerks C/C++ v4.2 and up
<i>H_PPCMW.41</i>	PowerPC	MetaWare High C/C++ v4.1 and up
<i>H_ARMRM.10</i>	ARM (ARM mode)	ARM Ltd. C/C++ ADS v1.0 and up
<i>H_ARMMW.410</i>	ARM (ARM mode)	MetaWare High C/C++ v4.1 and up
<i>H_ARBRM.10</i>	ARM (Thumb mode)	ARM Ltd. C/C++ ADS v1.0 and up
<i>H_ARBMW.410</i>	ARM (Thumb mode)	MetaWare High C/C++ v4.1 and up
<i>H_M32MW.430</i>	MIPS32	MetaWare High C/C++ v4.3e

Figure 3.2-2 KwikPeg Compiler Configuration Header Files

## Library Specification File

The librarian for toolset *xxx* uses a **library specification file** *KP\_LIB.LBM* to create the KwikPeg Library. All library specification files are located in directory *TOOLUU\CC\_LBM* in the KwikPeg installation directory *KPGnnn*. These files follow the same naming convention used for compiler configuration header files. Hence, for processor *ttt*, toolset *xxx* and compiler version *vvv* or later, use specification file *L\_tttXXX.vvv*.

For every KwikPeg compiler configuration header file *H\_tttXXX.vvv* listed in Figure 3.2-2, there is a corresponding library specification file *L\_tttXXX.vvv* in directory *TOOLUU\CC\_LBM*.

When using one of these files, you must copy the file and rename it *KP\_LIB.LBM*.

## Tailoring Files

The KwikPeg Library can be constructed using your make utility, C/C++ compiler and object module librarian. When using a make utility to create the KwikPeg Library, a file which KADAK calls a **tailoring file** is used to tailor the library construction process for a particular C/C++ compiler and object librarian. Separate tailoring files are available for each toolset combination which KADAK supports. These tailoring files are provided ready for use with Microsoft's *NMAKE* utility.

Tailoring files follow the same naming convention used for compiler configuration header files and library specification files. Tailoring file *M\_tttXXX.vvv* can be used with Microsoft's *NMAKE*.

The mnemonic *ttt* identifies the target processor family. The mnemonic *xxx* identifies the compiler vendor. The file extension *vvv* identifies the first version of the compiler for toolset *xxx* with which the tailoring file was tested. The tailoring file can be used with subsequent versions of the tools until some change in their method of operation requires a tailoring file update.

The KwikPeg tailoring files *M\_tttXXX.vvv* listed in Figure 3.2-3 have been created and tested with Microsoft *NMAKE* for use with KwikPeg and the AMX kernel. The files will be found in directory *TOOLUU\TF\_MSOFTE* in the KwikPeg installation directory *KPGnnn*. The KwikPeg product is always delivered to you with the most recent available set of these tailoring files.

When using one of these files, you must copy the file and rename it *KPZZZCC.INC*.

<b>Tailoring File</b>	<b>Target Processor</b>	<b>Compiler</b>
<i>M_86MC.15</i>	80x86 (real mode)	Microsoft 16-bit Visual C/C++ v1.5, v1.52
<i>M_86TC.50</i>	80x86 (real mode)	Borland 16-bit C/C++ v5.0
<i>M_86PD.50</i>	80x86 (real mode)	Paradigm 16-bit C/C++ v5.0, v6.0
<i>M_86WC.110</i>	80x86 (real mode)	WATCOM 16-bit C/C++ v11.0
<i>M_386PD.60</i>	80x86 (protected mode)	Paradigm 32-bit C/C++ v5.0, v6.0
<i>M_68KDA.42</i>	68000	Diab-SDS C/C++ v4.2, v4.3
<i>M_68KIM.92</i>	68000	TASKING (Intermetrics) C/C++ v9.2r0
<i>M_68KME.20</i>	68000	Metrowerks C/C++ v2.0
<i>M_68KME.32</i>	68000	Metrowerks C/C++ v3.2
<i>M_68KMR.51</i>	68000	Mentor Graphics (Microtec) C/C++ v5.1
<i>M_CFDA.42</i>	ColdFire	Diab-SDS C/C++ v4.2, v4.3
<i>M_CFME.25</i>	ColdFire	Metrowerks C/C++ v2.5
<i>M_CFME.32</i>	ColdFire	Metrowerks C/C++ v3.2, v4.0
<i>M_PPCDA.41</i>	PowerPC	Diab-SDS C/C++ v4.1, v4.2, v4.3
<i>M_PPCME.42</i>	PowerPC	Metrowerks C/C++ v4.2
<i>M_PPCME.50</i>	PowerPC	Metrowerks C/C++ v5.0
<i>M_PPCMW.41</i>	PowerPC	MetaWare High C/C++ v4.1, v4.3
<i>M_ARMMW.410</i>	ARM	MetaWare High C/C++ v4.1
<i>M_ARMMW.420</i>	ARM	MetaWare High C/C++ v4.2
<i>M_ARMRM.10</i>	ARM	ARM Ltd. C/C++ ADS v1.0, v1.1
<i>M_ARBXXX.vvv</i>	Thumb	(see vendors <i>M_ARMXXX.vvv</i> shown above)
<i>M_M32MW.430</i>	MIPS32	MetaWare High C/C++ v4.3e

Figure 3.2-3 KwikPeg Tailoring Files

## Getting Ready

Before creating the KwikPeg Library, you must pick the proper files for use with your C/C++ compiler and object module librarian (archiver). Be aware that KADAK has observed that not all compilers operate correctly with every version of the Microsoft make utility. If the make process inexplicably fails, it will most frequently be because of incompatibilities between these tools.

From directory *TOOLUU\CC\_H*, pick the compiler configuration **header file** *H\_tttXXX.vvv* which matches your choice of toolset and compiler version. Copy that file into directory *TOOLUU* but with name *KPZZZCC.H*. You may have to overwrite the default copy created in that directory when KwikPeg was installed.

From directory *TOOLUU\CC\_LBM*, pick the **library specification file** *L\_tttXXX.vvv* which matches your choice of toolset and compiler version. Copy that file into directory *TOOLUU* but with name *KP\_LIB.LBM*. You may have to overwrite the default copy created in that directory when KwikPeg was installed.

From directory *TOOLUU\TF\_MSOFTE*, pick the **tailoring file** *M\_tttXXX.vvv* which matches your choice of toolset and compiler version. Copy that file into directory *TOOLUU* but with name *KPZZZCC.INC*. You may have to overwrite the default copy created in that directory when KwikPeg was installed.

## KwikPeg Library Make File

The KwikPeg Configuration Builder is used to create and edit your Library Parameter File, say *GUI\_LIB.UP*. It is this file which describes the KwikPeg options and features which your application requires. From this parameter file, the Configuration Builder generates the KwikPeg Library Make File, say *GUI\_LIB.MAK*.

The KwikPeg Library Make File *GUI\_LIB.MAK* is a make file which can be used to create the KwikPeg Library tailored to your specifications. This make file is suitable for use with Microsoft's *NMAKE* utility.

## Gathering Files

The block diagram in Figure 3.2-1 summarizes the components which are required to build the KwikPeg Library. All of these files must be present in the appropriate KwikPeg installation directories prior to making the KwikPeg Library. Each of the following source files must be present in the indicated directory.

Source File	Directory	File Purpose
<i>GUI_LIB.UP</i>	<i>MAKE</i>	KwikPeg Library Parameter File
<i>GUI_LIB.MAK</i>	<i>MAKE</i>	KwikPeg Library Make File
<i>KPZZZCC.INC</i>	<i>TOOLUU</i>	Tailoring File (for use with make utility)
<i>KPZZZCC.H</i>	<i>TOOLUU</i>	Compiler Configuration Header File
<i>KP_LIB.LBM</i>	<i>TOOLUU</i>	KwikPeg Library Specification File
		PEG video device driver for video controller as specified in Library Parameter File <i>GUI_LIB.UP</i>
<i>xxxxxxxx.HPP</i>	<i>PEG\INCLUDE</i>	Screen driver header file
<i>xxxxxxxx.CPP</i>	<i>PEG\SOURCE</i>	Screen driver source code

## Creating the KwikPeg Library

The KwikPeg Library must be constructed from within directory *MAKE* in the KwikPeg installation directory. Your Library Parameter File, say *GUI\_LIB.UP*, and your KwikPeg Library Make File, say *GUI\_LIB.MAK*, must be present in the KwikPeg *MAKE* directory.

All of the compilers and librarians used at KADAK were tested on a Windows<sup>®</sup> NT workstation. However, you can build the library using any recent version of Windows, provided that your software development tools operate on that platform.

To create the KwikPeg Library, proceed as follows. From the Windows NT Start menu, choose the MS-DOS Command Prompt from the Programs folder. Make KwikPeg installation directory *KPGnnn\MAKE* the current directory.

To use Microsoft's *NMAKE* utility, issue the following command.

```
NMAKE -fGUI_LIB.MAK "TOOLSET=XXX" "PEGPATH=pegpath"  
"OSPATH=amxpath" "KPF=GUI_LIB.UP"
```

The make symbol *TOOLSET* is defined to be the toolset mnemonic *XXX* used by KADAK to identify the software tools which you are using.

The symbol *PEGPATH* is defined to be the string *pegpath*, the full path (or the path relative to directory *KPGnnn\MAKE*) to your PEG installation directory.

The symbol *OSPATH* is defined to be the string *amxpath*, the full path (or the path relative to directory *KPGnnn\MAKE*) to your AMX installation directory.

The make symbol *KPF* is defined to identify the name of the Library Parameter File *GUI\_LIB.UP* from which the KwikPeg Library Make File *GUI\_LIB.MAK* was generated. Both of these files must be present in the KwikPeg *KPGnnn\MAKE* directory.

For example, assume that the PEG release from Swell Software, Inc. has been installed in directory *C:\PEG* and that AMX 68000 (part number PN532-1) has been installed in directory *C:\KADAK\AMX532*. Then, to build the KwikPeg Library using Microsoft's *NMAKE* utility and Metrowerks tools, issue the following command.

```
NMAKE -fGUI_LIB.MAK "TOOLSET=ME" "PEGPATH=C:\PEG"  
"OSPATH=C:\KADAK\AMX532" "KPF=GUI_LIB.UP"
```

By default, the KwikPeg Library will be created in directory *KPGnnn\LIB*. You can force the library to be created elsewhere by defining symbol *KPGLIB=libpath* on the make command line. The string *libpath* is the full path (or the path relative to directory *KPGnnn\MAKE*) to the directory in which you wish the library to be created.

**Note:** If you are using any of the following versions of AMX, you must also add the following symbol definition to the make command line.

AMX 4-ARM	<i>"AMX4ARM=1"</i>
AMX 386/ES	<i>"AMX386ES=1"</i>
AMX 386/EP	<i>"AMX386EP=1"</i>

## Generated KwikPeg Library Modules

All KwikPeg source files will be compiled and the resulting object modules will be placed in directory *KPGnnn\LIB*. The KwikPeg Library will be created from these object files and placed in directory *KPGnnn\LIB*. Note that the library file extension will be *.A* or *.LIB* or some other extension as dictated by the toolset which you are using.

*KP\_LIB.A*                      KwikPeg Library

In addition to the library module and the object modules used to create them, the following files will also be created in directory *KPGnnn\LIB*.

*KP\_LIB.UP*                      KwikPeg Library Parameter File  
*KP\_LIB.MAK*                    KwikPeg Library Make File  
*KP\_LIB.H*                        KwikPeg Library Header File

File *KP\_LIB.UP* is a copy of the Library Parameter File *GUI\_LIB.UP* which you identified on your make command line. It is copied to the *LIB* directory so that you have a record of the parameters used to produce the library present in the directory.

File *KP\_LIB.MAK* is a KwikPeg Library Make File which can be used to reproduce the library. It is generated in the *LIB* directory so that you have a record of the make file used to produce the library present in the directory. This file is derived from the KwikPeg Library Make Template file *KPnnnLIB.MT* and the parameters in Library Parameter File *KP\_LIB.UP*. It should match the make file *GUI\_LIB.MAK* which you passed to your make utility to start the make process.

File *KP\_LIB.H* is the KwikPeg Library Header File, a C header file generated by the make process. This file is derived from the KwikPeg Library Header Template file *KPnnnLIB.HT* and the parameters in Library Parameter File *KP\_LIB.UP*.

A copy of header file *KP\_LIB.H* will also be found in the *PEG\INCLUDE* directory. The make process copies the file there so that it is available for inclusion in the compilation of all C/C++ files in the library.

A copy of the toolset dependent header file *TOOLUU\KPZZZCC.H* will also be found in the *PEG\INCLUDE* directory. The make process copies the file there so that it is also available for inclusion in the compilation of all C/C++ files in the library.

### 3.3 Compiling Application Modules

Any application module which uses KwikPeg services must include the KwikPeg Library header file *KP\_LIB.H*. It is this module which provides access to the KwikPeg header files found in PEG installation directory *PEG\INCLUDE*. You must define the path to the PEG installation directory *PEG\INCLUDE* using compiler switches or an environment variable.

Your application can include the PEG header file *PEG.HPP* instead of *KP\_LIB.H*. Either of these files will include the other in order to properly compile your C++ source file.

In order to compile an application C++ source file, say *MYFILE.CPP*, which makes use of KwikPeg services, the following KwikPeg header files must also be present in the PEG installation directory *PEG\INCLUDE*.

<i>KP_LIB.H</i>	KwikPeg Library Header File
<i>KPZZZCC.H</i>	KwikPeg compiler specific definitions

Header file *KP\_LIB.H* is a copy of your KwikPeg Library Header File from your KwikPeg library directory. This file is created as a byproduct of the KwikPeg Library construction process described in Chapter 3.2 and is automatically copied to directory *PEG\INCLUDE*.

Header file *KPZZZCC.H* is the compiler specific file which you selected and installed in KwikPeg installation directory *TOOLUU*. When you build your KwikPeg Library as described in Chapter 3.2, this file is automatically copied to directory *PEG\INCLUDE*.

If source file *MYFILE.CPP* makes calls to AMX service procedures, you must also have access to all of the required AMX header files.

You must also have access to your C/C++ library header files so that KwikPeg header files can reference them.

With these header files in place, your application module *MYFILE.CPP* is ready to be compiled. Simply follow the procedure described in the toolset specific chapter of the AMX Tool Guide.

#### Note

Your application can include the PEG header file *PEG.HPP* instead of *KP\_LIB.H*. Either of the files will include the other in order to properly compile your C++ source file.

### 3.4 Linking the Application

To add KwikPeg to your application, the KwikPeg Library must be added to your link specification.

```
    KP_LIB.A          KwikPeg Library
```

Note that object file extensions may be `.O` or `.OBJ` and library file extensions may be `.A` or `.LIB`. Either may be some other extension as dictated by the toolset which you are using.

The KwikPeg Library must always be included in the link. The KwikPeg Library must precede the AMX libraries in the link sequence. If you are using the KwikNet TCP/IP Stack, the KwikPeg Library must precede all KwikNet libraries.

Although every effort has been made to ensure that each module in a KwikPeg Library contains only forward references to other modules in the same library, the goal has proved impossible to attain. Hence, some library modules do include backward references. This characteristic requires that the libraries be searched recursively until all resolvable references have been satisfied. Most linkers will meet this requirement. If yours does not, you will be forced to include the KwikPeg Library more than once in your link specification.

There is little difference in linking an AMX application with or without KwikPeg. Instructions for linking an AMX system are provided in the toolset specific chapters of the AMX Tool Guide.

#### **Important !!!**

When using KwikPeg, you must link with your compiler's C/C++ runtime library, not the regular C Library specified in the AMX Tool Guide. Be sure to select the library with the correct endianness for your target processor.

## 3.5 Integrating KwikPeg with AMX

### 3.5.1 AMX System Configuration

KwikPeg includes its own interface to the underlying AMX operating system. The KwikPeg OS Interface for AMX is ready for use without modification or customization.

KwikPeg makes few demands for AMX resources. Consequently, there are few changes to your AMX System Configuration Module required to accommodate KwikPeg.

#### KwikPeg Task

A single KwikPeg Task drives the KwikPeg GUI. You must add this task to your list of predefined tasks in your AMX System Configuration Module. You can use the AMX Configuration Builder to do so. The task's definition is as follows:

Tag	<i>KPEG</i>
Procedure name	<i>kp_task</i>
Task Id Variable	<i>kptskid</i>
Priority	Usually low
Task stack size	AMX minimum plus 4096 (adjust as needed)
Queue 0	0
Queue 1	0
Queue 2	0
Queue 3	0

With AMX 86 and some toolsets, you may have to add a leading or trailing underscore ( `_` ) character to the task procedure name.

The task stack size requirement will vary with the particular version of AMX you are using. As a good rule of thumb, choose a stack size which is approximately 4096 bytes more than the minimum stack size required for an AMX task. Add more stack if any of the following conditions exist.

- Your target is a RISC processor with increased stack demands.
- KwikPeg options are used which make use of file system services.

For all versions of AMX, the KwikPeg Task is a simple task with no AMX message queues. KwikPeg uses private messaging blocks for internal communication with the KwikPeg Task. The number of available messaging blocks is defined by you on the Target property page when you create your KwikPeg Library Parameter File. You may have to increase the number of messaging blocks if any of the following conditions exist.

- You have many tasks generating graphics in separate windows.
- You have many sources of signals which initiate a display update.
- You are using the PEG multithread execution model.

## **AMX Interrupt Stack**

You may have to grow the size of your AMX Interrupt Stack, the stack used by all Interrupt Service Procedures. The stack must be large enough to meet the needs of each of the KwikPeg device drivers which service your physical video display and input devices. As a good rule of thumb, choose a stack size which is approximately 500 bytes more than the minimum required AMX Interrupt Stack size.

## **KwikPeg Semaphores**

KwikPeg requires a minimum of four semaphores for its operation. These semaphores will be created dynamically by KwikPeg during its initialization phase. You must include the AMX Semaphore Manager in your AMX System Configuration Module by declaring a requirement for at least four (4) semaphores.

If you configure KwikPeg to use standard C for memory allocation, an additional semaphore will be needed. If you use a file system other than AMX/FS and require file access locking, allocate one more semaphore.

If you use the PEG multithread execution model, you must also allocate one additional semaphore for each GUI Task that operates in parallel with the PEG Presentation Manager.

## **KwikPeg Memory Pool**

If you configure KwikPeg to use AMX memory management services, one AMX memory pool will be required. The memory pool will be created dynamically by KwikPeg during its initialization phase. You must include the AMX Memory Manager in your AMX System Configuration Module by declaring a requirement for at least one memory pool. The memory for the pool must be allocated by you. Use the KwikPeg Configuration Builder to edit your KwikPeg Library Parameter File and select one of the memory assignment techniques specified on the OS property page.

## **KwikPeg Timer**

KwikPeg requires one AMX timer for its operation. This timer will be created dynamically by KwikPeg during its initialization phase. You must include the AMX Timer Manager in your AMX System Configuration Module by declaring a requirement for at least one timer. Of course, to support timing you must also include an AMX Clock Handler as part of your application.

The KwikPeg timer operates at the KwikPeg clock frequency defined by you in your KwikPeg Library Parameter File. The period of this clock must correspond to an integer multiple of AMX system ticks. For example, you may have a hardware clock interrupt frequency of 1KHz with an AMX tick frequency of 100 Hz and a KwikPeg clock frequency of 10 Hz. In this case, the KwikPeg timer will be serviced once every 10 AMX ticks providing service at 100 ms intervals.

## KwikPeg Restart and Exit Procedures

KwikPeg procedure *kp\_osready()* **must be included** in your list of Restart Procedures in your AMX System Configuration Module. It is recommended that this procedure be first in the list of Restart Procedures. This procedure creates the semaphores used by KwikPeg and prepares the memory allocation subsystem for use by KwikPeg and your C++ modules.

KwikPeg includes a startup procedure *kp\_enter()* and a shutdown procedure *kp\_exit()*. You can include the KwikPeg startup procedure *kp\_enter()* in your list of Restart Procedures in your AMX System Configuration Module. It is this procedure which starts the KwikPeg Task to initialize the KwikPeg GUI. Alternatively, one of your own Restart Procedures can call *kp\_enter()*. The position of this procedure in the list of Restart Procedures is not critical since KwikPeg services must not be used by tasks until they are certain that the KwikPeg Task is ready, as determined by a call to KwikPeg procedure *kp\_state()*.

Another approach is to have a task call *kp\_enter()* to start KwikPeg. Again, no task must use KwikPeg services without first calling KwikPeg procedure *kp\_state()* to ascertain that KwikPeg is ready.

If your AMX application allows an orderly shutdown and exit from AMX, you can add the KwikPeg shutdown procedure *kp\_exit()* to your list of Exit Procedures in your AMX System Configuration Module. Alternatively, one of your own Exit Procedures can call *kp\_exit()*. Insert this procedure into the list at the point in the exit sequence at which the KwikPeg GUI is no longer required. You must ensure that all tasks have stopped using KwikPeg services before you allow KwikPeg to shut down.

### Important !!!

KwikPeg procedure *kp\_osready()* **must be included** in your list of Restart Procedures in your AMX System Configuration Module.

### Note

All KwikPeg application service procedures are documented under the topic "Getting Started" which can be reached from the "Index" page of the on-line KwikPeg Reference Manual.

## **AMX 86 PC Supervisor**

AMX 86 includes a component called the PC Supervisor which permits this version of AMX to be used with DOS on PC platforms. Special care must be taken when using the PC Supervisor with AMX and KwikPeg.

The PC Supervisor's Clock Tick Task and Keyboard Task must be of higher priority than the KwikPeg Task to ensure that they operate without interference from GUI activity.

The PC Supervisor Task must be of lower priority than the KwikPeg Task and all application tasks which use KwikPeg so that DOS operations do not interfere with their use of GUI services.

These task prioritization rules work provided that tasks which use KwikPeg services never go compute bound. For example, if a task continuously polls a mouse interface to test for mouse motion, then any higher priority task which attempts to use DOS services will appear to hang because the low priority PC Supervisor Task is unable to execute to service the DOS request. In such cases, you will have no choice but to raise the priority of the PC Supervisor Task and accept the fact that DOS operations can temporarily block tasks of lower priority.

## **AMX 86 Device Drivers**

There are no device driver dependent alterations of your AMX 86 System Configuration Module needed to use any KwikPeg keyboard or mouse device driver.

The device driver's initialization procedure will create an ISP root and install the pointer to that ISP root into the AMX Vector Table.

You may have to edit the keyboard or mouse device driver source module to alter the AMX 86 interrupt vector number assigned to the device for use with your particular target hardware.

### 3.5.2 AMX Target Configuration

The KwikPeg video device drivers are not interrupt driven unless modified by you for such use. The KwikPeg keyboard and mouse device drivers can be interrupt driven or polled. When used with interrupts, the input device drivers include an AMX Interrupt Service Procedure (ISP) consisting of two (sometimes three) parts. All such drivers require an ISP root and an Interrupt Handler. Some versions of AMX also require the driver to provide an ISP stem.

**Note**

You may have to edit the keyboard or mouse device driver source module to alter the AMX interrupt vector number assigned to the device for use with your particular target hardware.

#### 16-Bit AMX Systems

AMX 86 does not utilize a Target Configuration Module.

#### 32-Bit AMX Systems

There must be one ISP definition for each device driver interrupt source. Each ISP definition identifies the names of the ISP root and the ISP Handler. It also provides the name of the ISP stem if required by AMX. Each ISP definition may include a device dependent pointer parameter. No interrupt vector number is required in the definition since each KwikPeg device driver installs the pointer to its ISP root into the AMX Vector Table when its initialization procedure is called.

For all 32-bit implementations of AMX, the ISP definition is presented in the AMX Target Configuration Module. The AMX Configuration Builder allows the Target Parameter File to be easily edited to add ISP definitions.

If you are using a KwikPeg device driver with a 32-bit version of AMX and want the driver to be interrupt driven, you must edit your AMX Target Parameter File to include a device driver definition with the following characteristics:

ISP root:	<i>kp_dd_root</i>	in AMX Target Configuration Module
ISP handler:	<i>kp_dd_handler</i>	in the device driver module
AMX vector:	<i>-1</i>	not used; device driver specifies vector
ISP parameter:	<i>none</i>	not used
Parameter type:	<i>0</i>	no parameter

The directive in your AMX Target Parameter File will appear as follows:

```
...ISPC kp_dd_root,kp_dd_handler,-1,,0
```

If you are using AMX PPC32 for the PowerPC, the device driver also includes an ISP stem. Hence, in addition to the parameters listed above, the device definition in your AMX Target Parameter File must also include the following parameter:

ISP stem:	<i>kp_dd_stem</i>	in the device driver module
-----------	-------------------	-----------------------------

The directive in your AMX PPC32 Target Parameter File will appear as follows:

```
...ISPC kp_dd_root,kp_dd_stem,kp_dd_handler,-1,,0
```

### 3.6 Constructing the KwikPeg Sample Program

Construction of any KwikPeg application for use with AMX will closely follow the steps needed to build the KwikPeg Sample Program. These steps are summarized below. The sample program files are located in KwikPeg installation directory *KPGnnn\SAMPLE*.

1. If you have not yet ported the AMX Sample Program to your hardware, you will have to do so now since elements of it are needed to build the KwikPeg Sample Program. In particular, the AMX clock driver is required for timing and the AMX serial UART driver is required for keyboard input from a terminal.
2. Copy your toolset dependent AMX Sample Program Link Specification File (*CJSAMPLE.LKS* or equivalent) and rename it *KPSAMPLE.LKS*. Edit the file and replace object file *CJSAMPLE.O*. (*AM831SAM.OBJ* for AMX 86) with the KwikPeg Sample Program object files *KPSAMPLE.O* and *KPSAMWIN.O*. Add the KwikPeg Library module *KP\_LIB.A* (or equivalent) to the link specification file immediately prior to the AMX Library module.

Replace the regular C Library specified in the link specification file with your compiler's C/C++ runtime library. Be sure to select the library with the correct endianness for your target processor.

3. If you wish to use your own AMX clock driver, edit the KwikPeg Sample Program User Parameter File and Target Parameter File (not required for AMX 86) to identify your clock driver and its frequency. Replace the AMX clock driver object module specified in Link Specification File *KPSAMPLE.LKS* with the object module for your clock driver.
4. If necessary, adapt the KwikPeg board driver *KP\_BOARD.C* to accommodate your target processor, device interfaces and interrupt management scheme. This module is located in KwikPeg installation directory *KPGnnn\PEGFILES\SOURCE*. Copy your revised file *KP\_BOARD.C* to the PEG installation directory *PEG\SOURCE* so that it will be properly incorporated into the KwikPeg Library in the next step.
5. Using the KwikPeg Configuration Builder, open the sample program's Library Parameter File *KPSAMLIB.UP* (see Chapter 2.3). On the Video property page, select a standard PEG video device driver or identify your own that you have created for use on your hardware. If you have created your own video device driver source file *MYVIDEO.CPP*, make sure that it is installed in PEG directory *PEG\SOURCE* and that your driver header file *MYVIDEO.HPP* is installed in directory *PEG\INCLUDE*. Use the builder to generate your KwikPeg Library Make File *KPSAMLIB.MAK*. Use this file, the Microsoft *NMAKE* utility, and your C/C++ compiler and librarian to generate the KwikPeg Library (see Chapter 3.2).

6. If you are using a **32-bit version of AMX**, proceed as follows.  
Otherwise, go to step 7.

Using the AMX Configuration Builder, open the sample program's User Parameter File *KPSAMSCF.UP* and generate the AMX System Configuration Module *KPSAMSCF.C*. Compile the module as described in the AMX Tool Guide for the toolset which you are using.

Using the AMX Configuration Builder, open the sample program's Target Parameter File *KPSAMTCF.UP* and generate the AMX Target Configuration Module *KPSAMTCF.S*. Assemble the module as described in the AMX Tool Guide for the toolset which you are using.

Go to step 8.

7. If you are using **AMX 86**, proceed as follows.  
Otherwise, go to step 8.

Using the AMX 86 Configuration Builder, open the sample program's User Parameter File *KPSAMSCF.UP* and generate the AMX System Configuration Module *KPSAMSCF.ASM*. Assemble the module as described in the AMX 86 Tool Guide for the toolset which you are using.

8. Compile the KwikPeg Sample Program application modules *KPSAMPLE.CPP* and *KPSAMWIN.CPP*. Compile these C++ modules with full debug information to improve your view when running the sample with your debugger.
9. Link the modules listed in your KwikPeg Sample Program Link Specification File *KPSAMPLE.LKS* together with your KwikPeg Library, the AMX Library and your C/C++ Library to create your KwikPeg application load module (see Chapter 3.4).
10. Use your debugger to load and execute the KwikPeg Sample Program.

### 3.7 An Alternate KwikPeg Library Building Method

If you are not able to use Microsoft's *NMAKE* utility to create your KwikPeg Library, there is an alternate library construction method that you can use. This alternate method may also be used if you are building your library within an Integrated Development Environment (IDE) provided by your C/C++ compiler vendor. The alternate method is also suitable for use in batch processing environments.

#### Required Files

The files listed below must be present in your PEG installation directory prior to making the KwikPeg Library. Each of the following source files must be present in the indicated directory.

Source File	Directory	File Purpose
<i>KPZZZCC.H</i>	<i>PEG\INCLUDE</i>	Compiler Configuration Header File
<i>GUI_LIB.UP</i>	<i>PEG\INCLUDE</i>	KwikPeg Library Parameter File
<i>KP_LIB.H</i>	<i>PEG\INCLUDE</i>	KwikPeg Library Specification File
		PEG video device driver for video controller as specified in Library Parameter File <i>GUI_LIB.UP</i>
<i>xxxxxxxx.HPP</i>	<i>PEG\INCLUDE</i>	Screen driver header file
<i>xxxxxxxx.CPP</i>	<i>PEG\SOURCE</i>	Screen driver source code

If you have not already done so, copy your **video device driver** header and source files to your PEG installation directories *C:\PEG\INCLUDE* and *C:\PEG\SOURCE* respectively.

File *KPZZZCC.H* is the **compiler configuration header file** for the software development tools that you are using. This is the same file used in the normal library make process which has been described in Chapter 3.2. From KwikPeg directory *KPGnnn\TOOLUU\CC\_H*, pick the compiler configuration header file *H\_tttXXX.vvv* which matches your choice of toolset and compiler version. Copy that file into PEG directory *PEG\INCLUDE* but with name *KPZZZCC.H*.

File *GUI\_LIB.UP* is the **Library Parameter File** which you created using the KwikPeg Configuration Builder as described in Chapter 2. It is this file which describes the KwikPeg options and features which your application requires. You can use the Configuration Builder to create this file and save it directly into directory *PEG\INCLUDE*.

File *KP\_LIB.H* is the **KwikPeg Library Header File**, a C header file which the Configuration Builder will generate for you from the parameters in your Library Parameter File *GUI\_LIB.UP*. To create this file, proceed as follows.

Use the Configuration Builder to open your *GUI\_LIB.UP* file. From the builder's File menu, pick the Templates... command. From the Module list, choose the KwikPeg Library Header File selector as illustrated in Figure 3.7-1. Edit the path for the Configuration Output File so that the file *KP\_LIB.H* is directed to your PEG installation directory, say *C:\PEG\INCLUDE*, as shown. Close the Templates dialog.

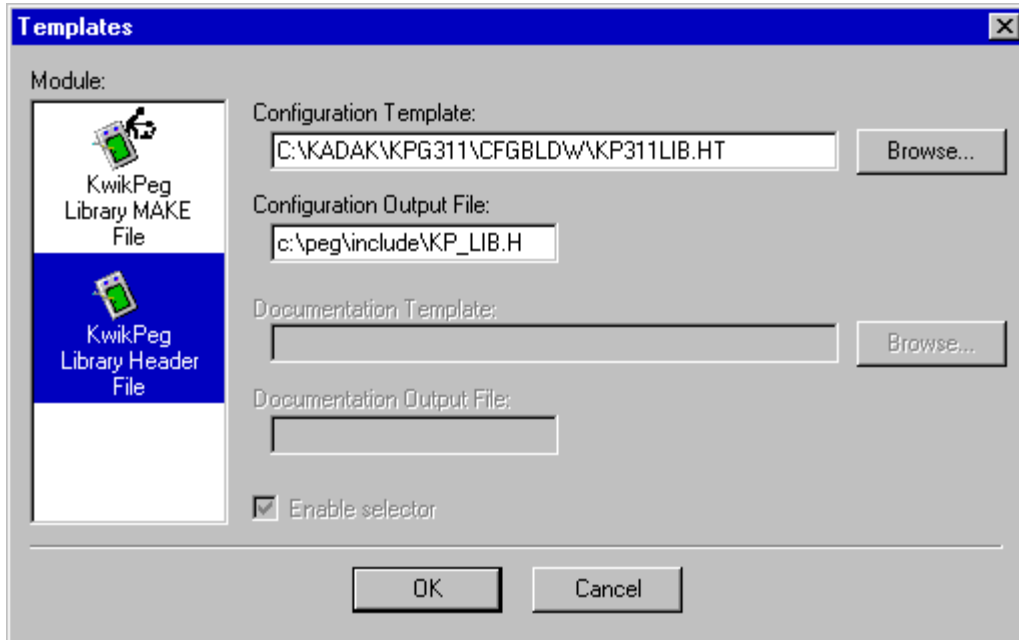


Figure 3.7-1 Setting the KwikPeg Library Header File Path

From the Module list on the builder's main screen, choose the KwikPeg Library Header File selector. Then click on the Generate button on the toolbar or choose the Generate... command from the File menu. The Configuration Builder will generate your KwikPeg Library Header File *KP\_LIB.H* and store it in PEG directory *C:\PEG\INCLUDE* so that it is available for inclusion in the compilation of all C/C++ files in the library.

## Making the KwikPeg Library

With all of the required PEG and KwikPeg header and source files present in the PEG installation directory, you are ready to build a KwikPeg Library. Check that all of the files in KwikPeg installation directory *KPGnnn\PEGFILES* are also present in the *PEG* directory. If any files are missing, copy them to the *PEG* directory.

From KwikPeg directory *KPGnnn\TOOLUU\CC\_LBM*, pick the **Library Specification File** *L\_tttXXX.vvv* which matches your choice of toolset and compiler version. This file has been described in Chapter 3.2. Examine the file carefully. It identifies all of the object modules which you must place into your KwikPeg Library. It also specifies the required order of the object modules in the library. In addition to these modules, you must add your video screen driver object module as the last module in the library.

From KwikPeg directory *KPGnnn\TOOLUU\CC\_MSOF*, pick the **Tailoring File** *M\_tttXXX.vvv* which matches your choice of toolset and compiler version. This file has been described in Chapter 3.2. Examine the file carefully to ensure that you are using the appropriate command line switches for the software development tools that you have indicated you are using to compile source files and create the library.

## A. KwikPeg Glossary

API	An application programming interface defines the method by which a software program can access software components such as procedures in the KwikPeg Library.
Bitblit	A screen drawing operation supported by some video controllers to rapidly move one region of the screen to another, thereby providing fast and smooth scrolling effects.
Bpp	Short form for bits per pixel. Used to define the number of bits of information used to identify the color of a pixel.
Child	A window or control object which is directly attached to another object. A child object is said to be descended from the object to which it is attached.
Clock Handler	The name given to the procedure which is called by the ISR or ISP root which services the hardware clock interrupt.
Clock Tick	The interrupt generated by a hardware clock.
Conforming ISP	An AMX Interrupt Service Procedure consisting of an ISP root which calls an Interrupt Handler which has the right to make calls to a subset of the KwikPeg service procedures.
Control	A GUI object that appears on the display screen, usually within the bounds of some window object. Controls are often characterized as buttons and text boxes that support user interaction. However, within KwikPeg, a control object can also be considered a window since it can have other objects attached to it, a feature usually associated with a window.
Current Focus	See Input Focus Branch.
Display	Any video or LCD device providing a screen which can be used for presenting graphical images.
Error Code	A series of signed integers used by KwikPeg to indicate error or warning conditions detected by KwikPeg service procedures.
Exit Procedure	An AMX or application procedure executed by AMX during the exit phase when an AMX system is shut down.
Fatal Error	A condition detected by KwikPeg which is considered so abnormal that to proceed might risk catastrophic consequences.

FIFO	First in, first out. Usually used to refer to the ordering of elements in a queue or linked list.
Focus	See Input Focus Branch.
GUI	The acronym for a Graphical User Interface.
Handle	An identifier assigned by AMX or KwikPeg for use by your application to reference a private AMX or KwikPeg data item.
HMI	The acronym for Human Machine Interface. The KwikPeg HMI classes include a collection of graphical objects which implement complex user interfaces such as dials and gauges.
Input Device	Any external device such as a keyboard, mouse, joystick or touch screen panel which is used interactively to present information, or requests for action, to the KwikPeg GUI.
Input Focus Branch	The collection of KwikPeg objects (windows and controls) in a branch of the KwikPeg presentation tree which have input focus at any particular instant. One of these objects, deemed the current input object, will receive notification when any input device initiates an action or presents information for processing.
Interrupt Handler	An application procedure called from an ISR or ISP root to service an interrupting device.
Interrupt Service Procedure (ISP)	A procedure in an AMX application which is executed in response to an external device interrupt request.
Interrupt Service Routine (ISR)	The procedure, in any application, which is executed in response to an external device interrupt request. In an AMX application, such a procedure is called an ISP.
ISP	See Interrupt Service Procedure
ISP root	The ISP code fragment (produced by the AMX Configuration Generator) which informs AMX that an interrupt has occurred and calls an application Interrupt Handler.
ISR	See Interrupt Service Routine
KwikPeg Task	The private KwikPeg procedure which is responsible for all timing control and GUI related event sequencing in a KwikPeg application.
LCD	Short form for Liquid Crystal Display. Any of a variety of flat screen display devices which employ LCD technology.
Leaf Node	Any KwikPeg object which has no children.

Library Parameter File	A text file which can be edited by the KwikPeg Configuration Builder to describe a particular KwikPeg Library configuration.
Memory Block	A portion of a memory pool that has been allocated for use by one or more tasks.
Memory Pool	A collection of memory sections whose use is controlled by the AMX Memory Manager.
Memory Pool Id	The handle assigned to a memory pool by AMX for use as a unique memory pool identifier.
Memory Section	A contiguous region of memory assigned to the AMX Memory Manager for allocation to application tasks.
Modal	A method of operation in which a window object requiring interaction with a user retains control (maintains input focus) until the window is closed. Most dialog windows are modal.
Multitasking	A method of program execution in which an operating system like AMX makes it appear as though several procedures (called tasks) are running concurrently.
KwikPeg Tick	A multiple of the system tick from which the fundamental KwikPeg unit of time is derived. All KwikPeg time intervals in the system are measured in multiples of the KwikPeg tick.
Object	Any instance of a C++ class. When used in the context of KwikPeg, the term object will most often be used to reference a window or control which forms part of a GUI.
OS	A short form for the two words operating system. When the term OS is used alone, no assumption about the operational characteristics of the OS can be made.
Parent	The window or control object to which one or more other objects, called child objects, are directly attached. The attached child objects are said to be descended from the parent object.
PEG	An acronym for Portable Embedded GUI, the name of the product from Swell Software, Inc. from which KwikPeg is derived.
Pixel	The unit of measurement used to locate the x-y coordinates of a point on the display screen surface. Also used to reference the color dot that is created at that position on the screen.
Pointing Device	Any external device such as a mouse, joystick or touch screen panel that is used interactively to identify and initiate some operation to be performed by the KwikPeg GUI.

Presentation Manager	The KwikPeg Presentation Manager is an invisible window object maintained and serviced by the KwikPeg Task. The Presentation Manager's window corresponds to the display screen drawing area.
Presentation Tree	The KwikPeg Presentation Manager maintains a tree-like structure which identifies all of the KwikPeg objects (windows and controls) which, at any instant, can be drawn on the display screen. The list of objects is called the presentation tree. Any object in the presentation tree may be fully visible, partly obscured or totally invisible.
RAD	An acronym for Rapid Application Development, a goal of all embedded systems developers.
RAM	Alterable memory used for data storage and stacks.
Restart Procedure	An AMX or application procedure executed by AMX during the initialization phase when an AMX system is started.
RLE Compression	A Run Length Encoding algorithm is used to compress a bitmap image to reduce its memory or disk storage requirements.
ROM	Read only memory of all types including PROMs, EPROMs and EAROMs.
RTOS	A short form for referencing a real-time operating system, usually one with multitasking capability and reasonably good response to rapidly occurring external events.
RT/OS	The KwikPeg syntax used to specify a general purpose operating system. The term RT/OS is used to encompass both multitasking and single threaded operating systems when the distinction is irrelevant.
Screen	The surface of a display containing all of the possible pixels that can be used for rendering a visible image.
Semaphore	A data structure which can be used by an RTOS to provide an event signaling mechanism or mutually exclusive access by tasks to specific resources.
Sibling	If multiple window or control objects are directly attached to a parent object, these child objects are collectively referred to as siblings. Each child object is a sibling of all other child objects having the same parent object.
Single threaded	A method of program execution in which all procedures execute sequentially. This method of operation is sometimes called single tasking. Although interrupts can cause a brief digression in the program's sequential operation, execution after an interrupt has been serviced always resumes in the procedure which was preempted by the interrupt.

System Configuration Module	A software module, produced by the AMX Configuration Builder, which defines the characteristics of a particular AMX application.
System Tick	A multiple of the hardware clock tick from which the fundamental unit of time for an RT/OS is derived. All time intervals in the system are measured in multiples of the system tick.
Tag	A 4-character name that can be assigned to any AMX system data structure when it is created. A tag can be used to find the identifier of a task, timer, semaphore, event group, mailbox, message exchange, memory pool or buffer pool with a particular name.
Tailoring file	A special make file included by the make specification file which is used to build a library or application. The tailoring file provides macro definitions and implicit rules which specify how the make utility can use a specific set of software development tools (compiler, assembler, librarian, linker/locator).
Target Configuration Module	A software module, produced by the AMX Configuration Builder, which defines the characteristics of your target hardware as used in a particular AMX application.
Task	An application procedure which is executed by an RTOS in a way which makes it look as though all such procedures are executing at once.
Task Id	The handle assigned to a task by KwikPeg for use as a unique task identifier.
Task Priority	The priority at which a task executes.
Timer	A facility provided by AMX to permit precise interval measurement in AMX applications.
Timer Id	The handle assigned to a timer by AMX for use as a unique timer identifier.
Timer Procedure	An application procedure which is executed by AMX whenever the corresponding timer interval expires.
Top Level	Any KwikPeg window or control which has been directly added to the invisible window maintained by the KwikPeg Presentation Manager is called a top level window.
Video	The technology used to generate a visible screen pixel from the numeric representation used for programming. The device that translates the numeric pixel information to color dots for delivery to a display screen is called a video controller.
Window	A GUI object that appears on the display screen, usually with one or more other window objects or control objects contained within its bounds. Window objects are commonly characterized by the title bars, status bars, scroll bars and menu bars which they sport.

This page left blank intentionally.

## B. KwikPeg Error Codes

PEG does not return error or warning status to the application. Hence, there are no KwikPeg error or warning codes defined. All non-recoverable error conditions are considered fatal for the reasons indicated by the fatal error codes listed below. To assist you during testing, the hexadecimal value of the least significant 16-bits of the error code is listed as it might appear in a register or memory dump.

### KwikPeg Fatal Error Codes

Mnemonic	Value (dec)	Value (hex)	Meaning
<i>KP_FERNOTASK</i>	100	0x0064	Cannot find KwikPeg Task
<i>KP_FERNOTMR</i>	101	0x0065	Cannot create KwikPeg timer
	102	0x0066	reserved
	to		
	109	0x006D	reserved
<i>KP_FERNOSEM4</i>	110	0x006E	No semaphores available for use
	111	0x006F	reserved
<i>KP_FERLOCK</i>	112	0x0070	Resource lock failed
<i>KP_FERUNLOCK</i>	113	0x0071	Resource unlock failed
<i>KP_FERNOMEM</i>	114	0x0072	No memory for allocation
<i>KP_FERBADMEM</i>	115	0x0073	Freeing memory that was never allocated
<i>KP_FEREXIT</i>	116	0x0074	Cannot complete shutdown of KwikPeg
	117	0x0075	reserved
	118	0x0076	reserved
<i>KP_FERPORT</i>	119	0x0077	Custom port panic
<i>KP_FERPANIC</i>	120	0x0078	KwikPeg GUI panic

This page left blank intentionally.

## C. KwikPeg Fonts

### C.1 KwikPeg Vector Fonts

KwikPeg vector fonts are described in the PEG Programming Manual. Figure C.1-1 is an illustration of a vector font as generated by the example provided in the KwikPeg installation directory `KPGnnn\PEG\EXAMPLES\VECFONT`. This screen shot was produced using the KwikPeg prototyping facility. The vector font will appear exactly as shown when rendered by KwikPeg on your target screen.

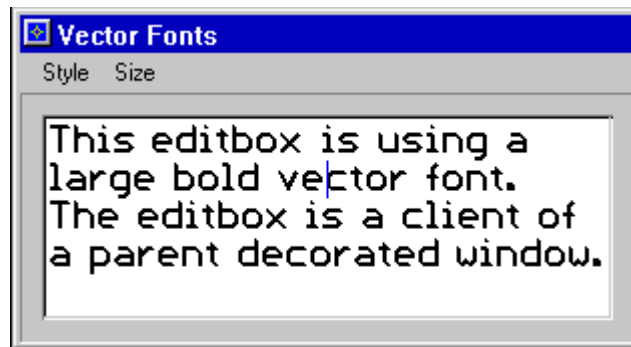


Figure C.1-1 KwikPeg Vector Font

## C.2 The KwikPeg System Font

The KwikPeg system font named *SysFont* is summarized in Figure C.2-1. The numeric values used to identify each font character are given in both decimal and hexadecimal form. The KwikPeg characters are drawn left justified in the character column. This illustration was generated using the KwikPeg prototyping facility. The fonts will appear exactly as shown when rendered by KwikPeg on your target screen.

dec	hex	dec	hex	dec	hex	dec	hex				
0	00	32	20	64	40	@	96	60	`		
1	01		33	21	!	65	41	A	97	61	a
2	02		34	22	"	66	42	B	98	62	b
3	03		35	23	#	67	43	C	99	63	c
4	04		36	24	\$	68	44	D	100	64	d
5	05		37	25	%	69	45	E	101	65	e
6	06		38	26	&	70	46	F	102	66	f
7	07		39	27	'	71	47	G	103	67	g
8	08		40	28	{	72	48	H	104	68	h
9	09		41	29	}	73	49	I	105	69	i
10	0A		42	2A	*	74	4A	J	106	6A	j
11	0B		43	2B	+	75	4B	K	107	6B	k
12	0C		44	2C	,	76	4C	L	108	6C	l
13	0D		45	2D	-	77	4D	M	109	6D	m
14	0E		46	2E	.	78	4E	N	110	6E	n
15	0F		47	2F	/	79	4F	O	111	6F	o
16	10		48	30	0	80	50	P	112	70	p
17	11		49	31	1	81	51	Q	113	71	q
18	12		50	32	2	82	52	R	114	72	r
19	13		51	33	3	83	53	S	115	73	s
20	14		52	34	4	84	54	T	116	74	t
21	15		53	35	5	85	55	U	117	75	u
22	16		54	36	6	86	56	V	118	76	v
23	17		55	37	7	87	57	W	119	77	w
24	18		56	38	8	88	58	X	120	78	x
25	19		57	39	9	89	59	Y	121	79	y
26	1A		58	3A	:	90	5A	Z	122	7A	z
27	1B		59	3B	;	91	5B	[	123	7B	{
28	1C		60	3C	<	92	5C	\	124	7C	
29	1D		61	3D	=	93	5D	]	125	7D	}
30	1E		62	3E	>	94	5E	^	126	7E	~
31	1F		63	3F	?	95	5F	_	127	7F	
128	80		160	A0		192	C0	À	224	E0	à
129	81		161	A1		193	C1	Á	225	E1	á
130	82		162	A2		194	C2	Â	226	E2	â
131	83		163	A3		195	C3	Ã	227	E3	ã
132	84		164	A4		196	C4	Ä	228	E4	ä
133	85		165	A5		197	C5	Å	229	E5	å
134	86		166	A6		198	C6	Æ	230	E6	æ
135	87		167	A7		199	C7	Ç	231	E7	ç
136	88		168	A8		200	C8	È	232	E8	è
137	89		169	A9		201	C9	É	233	E9	é
138	8A		170	AA		202	CA	Ê	234	EA	ê
139	8B		171	AB		203	CB	Ë	235	EB	ë
140	8C		172	AC		204	CC	Ì	236	EC	ì
141	8D		173	AD		205	CD	Í	237	ED	í
142	8E		174	AE		206	CE	Î	238	EE	î
143	8F		175	AF		207	CF	Ï	239	EF	ï
144	90		176	B0		208	D0	Ð	240	F0	ð
145	91		177	B1		209	D1	Ñ	241	F1	ñ
146	92		178	B2		210	D2	Ò	242	F2	ò
147	93		179	B3		211	D3	Ó	243	F3	ó
148	94		180	B4		212	D4	Ô	244	F4	ô
149	95		181	B5		213	D5	Õ	245	F5	õ
150	96		182	B6		214	D6	Ö	246	F6	ö
151	97		183	B7		215	D7	×	247	F7	×
152	98		184	B8		216	D8	Ø	248	F8	ø
153	99		185	B9		217	D9	Ù	249	F9	ù
154	9A		186	BA		218	DA	Ú	250	FA	ú
155	9B		187	BB		219	DB	Û	251	FB	û
156	9C		188	BC		220	DC	Ü	252	FC	ü
157	9D		189	BD		221	DD	Ý	253	FD	ý
158	9E		190	BE		222	DE	Þ	254	FE	þ
159	9F		191	BF		223	DF	ß	255	FF	ÿ

- Notes:**
1. The *NUL* character (value 0, *0x00*) does not display. Its width is zero pixels.
  2. The space character (value 32, *0x20*) is 4 pixels wide.
  3. The wide space character (value 160, *0xA0*) is 9 pixels wide.
  4. Undefined characters are 4 pixels wide (see codes 1-31).

Figure C.2-1 The KwikPeg System Font

### C.3 The KwikPeg Menu Font

The KwikPeg menu font named *MenuFont* is summarized in Figure C.3-1. The numeric values used to identify each font character are given in both decimal and hexadecimal form. The KwikPeg characters are drawn left justified in the character column. This illustration was generated using the KwikPeg prototyping facility. The fonts will appear exactly as shown when rendered by KwikPeg on your target screen.

dec	hex	dec	hex	dec	hex	dec	hex				
0	00	32	20	64	40	@	96	60	`		
1	01		33	21	!	65	41	A	97	61	a
2	02		34	22	"	66	42	B	98	62	b
3	03		35	23	#	67	43	C	99	63	c
4	04		36	24	\$	68	44	D	100	64	d
5	05		37	25	%	69	45	E	101	65	e
6	06		38	26	&	70	46	F	102	66	f
7	07		39	27	'	71	47	G	103	67	g
8	08		40	28	(	72	48	H	104	68	h
9	09		41	29	)	73	49	I	105	69	i
10	0A		42	2A	*	74	4A	J	106	6A	j
11	0B		43	2B	+	75	4B	K	107	6B	k
12	0C		44	2C	,	76	4C	L	108	6C	l
13	0D		45	2D	-	77	4D	M	109	6D	m
14	0E		46	2E	.	78	4E	N	110	6E	n
15	0F		47	2F	/	79	4F	O	111	6F	o
16	10		48	30	0	80	50	P	112	70	p
17	11		49	31	1	81	51	Q	113	71	q
18	12		50	32	2	82	52	R	114	72	r
19	13		51	33	3	83	53	S	115	73	s
20	14		52	34	4	84	54	T	116	74	t
21	15		53	35	5	85	55	U	117	75	u
22	16		54	36	6	86	56	V	118	76	v
23	17		55	37	7	87	57	W	119	77	w
24	18		56	38	8	88	58	X	120	78	x
25	19		57	39	9	89	59	Y	121	79	y
26	1A		58	3A	:	90	5A	Z	122	7A	z
27	1B		59	3B	;	91	5B	[	123	7B	{
28	1C		60	3C	<	92	5C	\	124	7C	
29	1D		61	3D	=	93	5D	]	125	7D	}
30	1E		62	3E	>	94	5E	^	126	7E	~
31	1F		63	3F	?	95	5F	_	127	7F	
128	80		160	A0		192	C0	À	224	E0	à
129	81		161	A1	¡	193	C1	Á	225	E1	á
130	82		162	A2	¢	194	C2	Â	226	E2	â
131	83		163	A3	£	195	C3	Ã	227	E3	ã
132	84		164	A4	¤	196	C4	Ä	228	E4	ä
133	85		165	A5	¥	197	C5	Å	229	E5	å
134	86		166	A6	¦	198	C6	Æ	230	E6	æ
135	87		167	A7	§	199	C7	Ç	231	E7	ç
136	88		168	A8	¨	200	C8	È	232	E8	è
137	89		169	A9	©	201	C9	É	233	E9	é
138	8A		170	AA	ª	202	CA	Ê	234	EA	ê
139	8B		171	AB	«	203	CB	Ë	235	EB	ë
140	8C		172	AC	¬	204	CC	Ì	236	EC	ì
141	8D		173	AD		205	CD	Í	237	ED	í
142	8E		174	AE	®	206	CE	Î	238	EE	î
143	8F		175	AF	¯	207	CF	Ï	239	EF	ï
144	90		176	B0	°	208	D0	Ð	240	FO	ð
145	91		177	B1	±	209	D1	Ñ	241	F1	ñ
146	92		178	B2	²	210	D2	Ò	242	F2	ò
147	93		179	B3	³	211	D3	Ó	243	F3	ó
148	94		180	B4	´	212	D4	Ô	244	F4	ô
149	95		181	B5	µ	213	D5	Õ	245	F5	õ
150	96		182	B6	¶	214	D6	Ö	246	F6	ö
151	97		183	B7	·	215	D7	×	247	F7	÷
152	98		184	B8	¸	216	D8	Ø	248	F8	ø
153	99		185	B9	¹	217	D9	Ù	249	F9	ù
154	9A		186	BA	º	218	DA	Ú	250	FA	ú
155	9B		187	BB	»	219	DB	Û	251	FB	û
156	9C		188	BC	¼	220	DC	Ü	252	FC	ü
157	9D		189	BD	½	221	DD	Ý	253	FD	ý
158	9E		190	BE	¾	222	DE	Þ	254	FE	þ
159	9F		191	BF	¿	223	DF	ß	255	FF	ÿ

- Notes:**
1. The *NUL* character (value 0, *0x00*) does not display. Its width is zero pixels.
  2. The space character (value 32, *0x20*) is 3 pixels wide.
  3. The wide space character (value 160, *0xA0*) is also 3 pixels wide.
  4. Undefined characters are 4 pixels wide (see codes 1-31).

Figure C.3-1 The KwikPeg Menu Font

This page left blank intentionally.

## D. KwikPeg Universal File System Interface

### D.1 Introduction

The KwikPeg GUI does not require a file system for normal use. However, the KwikPeg Image Converter can provide runtime conversion of image files to KwikPeg bitmap form if a file system is available. KwikPeg provides its own Universal File System (UFS) interface to the file services provided by the file system operating on the target system.

The Universal File System interface provides access to one of the following file systems:

- AMX/FS File System for use with the AMX Multitasking Kernel
- Standard C using the MS-DOS<sup>®</sup> file system
- Custom user defined file system
- KwikNet Universal File System used with the KwikNet TCP/IP Stack

AMX/FS can be used with AMX and KwikPeg on all of the supported target processors. Access to the MS-DOS file system is only available when using AMX 86 on a PC compatible platform. A custom file system can also be adapted for use with AMX and KwikPeg. Of course, AMX and KwikPeg can be used with the KwikNet Universal File System.

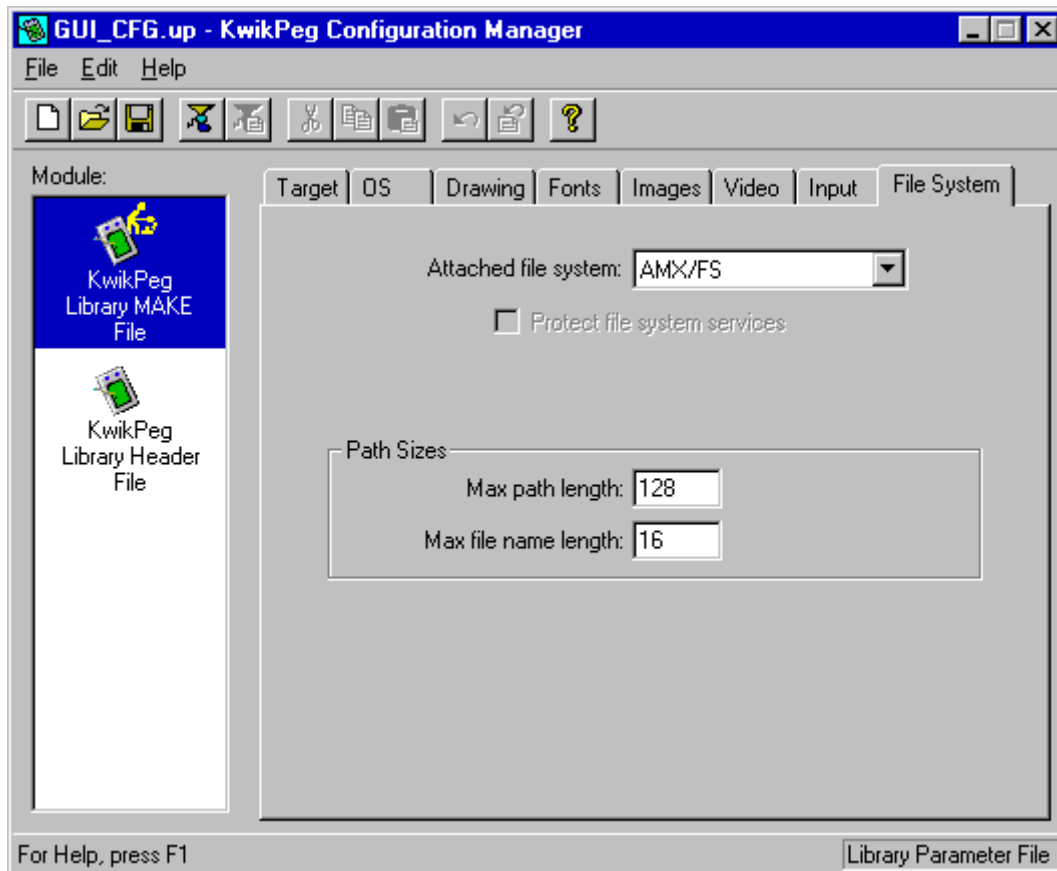
There should be no need to become familiar with the internal operation of the Universal File System unless you must adapt it for use with your own custom file system. This customization procedure is described in Appendix D.5.

If you are using KwikPeg with KADAK's KwikNet TCP/IP Stack, you may have to use the file system services provided by KwikNet. If you are using any of the optional KwikNet components such as FTP or the Web Server, then the KwikNet Universal File System must be used with KwikPeg to avoid file system conflicts.

## D.2 KwikPeg File System Parameters

The KwikPeg Universal File System (UFS) interface forms part of the KwikPeg Library. To include the UFS interface in the library, you must use the KwikPeg Configuration Builder to edit your KwikPeg Library Parameter File. This process, described in Chapter 2.3, is repeated in this appendix for completeness.

The Universal File System parameters are edited using the File System property page. The layout of the window is shown below.



## **Universal File System Parameters (continued)**

### **Attached File System**

From the pull down list, choose the type of underlying file system to which the Universal File System must connect. If you are not using any KwikPeg features which require a file system, select option None.

If you are using KwikPeg with the KwikNet TCP/IP Stack, you can use the KwikNet Universal File System if it has been included as part of your KwikNet configuration. To do so, select option KwikNet UFS from the pull down list.

### **Protect File System Services**

When operating in a multitasking environment, the file system services must be thread-safe. If the file system you have chosen to use is safe, leave this box unchecked. Otherwise, check this box and KwikPeg will use its file system locking mechanism to protect access to the unsafe file system services.

### **Maximum Path and File Name Lengths**

Specify the maximum number of characters which can appear in a path name string used to reference a directory location. The path length must include room for a terminating '\0' character. The path length excludes any file name.

Specify the maximum number of characters which can appear in a file name string used to identify any file. The file name includes the base name and extension(s) and any separating characters. The file name length must include room for a terminating '\0' character. The file name length excludes any path information.

There is no reason to set the path or file name lengths any greater than the maximum allowed by the selected file system. For AMX/FS and MS-DOS file systems, a path length of 128 and a file name length of 16 will be adequate.

To minimize memory waste in embedded applications in which short paths are the norm, the maximum path length can be reduced.

## D.3 Using the AMX/FS File System

The KwikPeg Universal File System (UFS) interface supports the AMX/FS File System. No customization is required. Any file devices, including RAM drives, hard drives, floppy drives and custom devices, which have been attached to the AMX/FS File System are accessible through the UFS interface.

To use the AMX/FS File System with KwikPeg, you must first install AMX and AMX/FS and test them in your intended target environment. The AMX/FS Sample Program offers a good starting point. Once you have it operating with your target hardware, you are ready to merge it with KwikPeg.

### AMX System Configuration

The first step is to merge the AMX configuration information required for both AMX/FS and KwikPeg into a new AMX configuration suitable for use with both products. You may have to increase the maximum number of tasks, timers and semaphores to meet the expanding requirements. Adjust the parameters in your AMX User Parameter File accordingly.

The KwikPeg Task must have file access. If you are using the graphics server model of execution, you may have other GUI tasks that also require access to images in files. Consequently, you may have to increase the maximum number of tasks which are permitted to concurrently use AMX/FS. You may also have to increase the maximum number of files which can be concurrently in use. Adjust the parameters in your AMX User Parameter File accordingly.

Your AMX configuration must include the device drivers for the KwikPeg video interface and input devices as well as for the AMX/FS file devices. You may have to adjust your choices of interrupt assignments to prevent conflicts among these devices.

Once you have an integrated AMX configuration, try using it to confirm that AMX/FS works well in the presence of KwikPeg but without KwikPeg actually in use. Then use the configuration to test that KwikPeg will operate with your display and input devices but without any file operations. Then you are ready to try KwikPeg and AMX/FS together.

## AMX System Startup

Special care must be taken when launching an AMX system which includes both KwikPeg and the AMX/FS File System. Initialize AMX/FS before starting KwikPeg. This implies that execution of AMX/FS Restart Procedure *fj\_restart()* must precede the call to KwikPeg function *kp\_enter()*.

AMX/FS requires that a logical drive be mounted before it can be accessed. This operation is not supported by the Universal File System interface. Hence, before starting the KwikPeg Task or any GUI tasks which require file support, your application must call AMX/FS procedure *fjdrvopen()* to mount each of the logical drives which these tasks are permitted to access.

There is no strict rule governing when logical drives should be mounted. The simplest solution is to have an application task unconditionally mount all available logical drives and then start KwikPeg. In this way, all KwikPeg components which require file support will have access to all logical drives whenever required.

## AMX System Shutdown

When shutting down your AMX application, you must stop KwikPeg before terminating AMX/FS. Hence KwikPeg function *kp\_exit()* must reach completion prior to the execution of AMX/FS Exit Procedure *fj\_exit()*.

## End of Line Indication

The use of CR (`'\r'`, ASCII `0x0D`), LF (`'\n'`, ASCII `0x0A`) or CRLF (CR followed by LF) as an end of line indicator in text files depends on the interpretation (translation) of strings by file streaming functions such as *fread()* and *fwrite()*.

Although the AMX/FS File System is MS-DOS<sup>®</sup> file format compatible, it does not provide a streaming level API. For example, the description of *fjopen()* states that all files are read and written in binary mode only. Hence AMX/FS does no translation of the data, even if the file is opened in text mode.

KwikPeg assumes that the underlying file system stores text files with CRLF as the end of line indicator. Such files can then be read and written as binary files.

Since MS-DOS stores text files with CRLF terminators, the KwikPeg interface is compatible with the most prevalent file system with which it must operate. Furthermore, since AMX/FS has no streaming support, CRLF has been adopted as the end of line indicator to be used by applications when recording text files with AMX/FS.

## D.4 Using the MS-DOS File System

When used with AMX 86, the KwikPeg Universal File System (UFS) interface supports the MS-DOS<sup>®</sup> file system. No customization is required. All file devices, including RAM drives, hard drives and floppy drives, are accessible through the UFS interface.

AMX 86 includes a component called the PC Supervisor which permits AMX to be used with MS-DOS on PC platforms. Special care must be taken when using the PC Supervisor with AMX and KwikPeg as described in Chapter 3.6.1.

The PC Supervisor Task must be of lower priority than tasks which use it to access MS-DOS. This requirement leads to the following recommended task priority order when using KwikPeg.

PC Supervisor Clock Tick Task	Highest priority
PC Supervisor Keyboard Task	
Application GUI Tasks	
KwikPeg Task	
PC Supervisor Task	Lowest priority

Note that the order of priority of tasks may have to be adjusted to reflect the relative importance of each of these services in your application.

## D.5 Using a Custom File System

The KwikPeg Universal File System (UFS) interface can be adapted to support a custom file system. To do so, you need only edit file *KPFSUSER.H* to meet the requirements of your custom file system. No other customization is required.

Once file *KPFSUSER.H* is ready, simply edit the File System parameters in your KwikPeg Library Parameter File to reference your custom file system. Then build your KwikPeg Library and link it with your application. The KwikPeg Universal File System interface will then use your custom file system for all file operations.

File *KPFSUSER.H* serves two purposes. As its name implies, it is a header file which maps all KwikPeg file access functions to those in your file system. However, it is also a code generating module which can provide a custom version of any file access function which is not available in your file system. The code generated by this module will actually reside in KwikPeg module *KP\_FILES.CPP* which is located in the KwikPeg installation directory *KPGnnn\PEG\SOURCE*.

The following minimal file system services must be provided by your file system.

- Open a file for read, write or read and write operation in binary mode
- Close a file
- Write *n* elements of size *m* to a file
- Read *n* elements of size *m* from a file
- Seek within a file
- Tell location of file pointer within a file
- Remove (delete) a file

This page left blank intentionally.