
# Getting Started with PersonalJava™ Solution for OS-9 (X86)

## Version 3.1

www.radisys.com

World Headquarters
5445 NE Dawson Creek Drive • Hillsboro, OR
97124 USA
Phone: 503-615-1100 • Fax: 503-615-1121
Toll-Free: 800-950-0044

International Headquarters
Gebouw Flevopoort • Televisieweg 1A
NL-1322 AC • Almere, The Netherlands
Phone: 31 36 5365595 • Fax: 31 36 5365620

RadiSys Microware Communications Software Division, Inc.
1500 N.W. 118th Street
Des Moines, Iowa 50325
515-223-8000

Revision A
October 2001

## Copyright and publication information

This manual reflects version 3.1 of PersonalJava Solution for OS-9.
Reproduction of this document, in part or whole, by any means, electrical, mechanical, magnetic, optical, chemical, manual, or otherwise is prohibited, without written permission from RadiSys Microware Communications Software Division, Inc.

## Disclaimer

The information contained herein is believed to be accurate as of the date of publication. However, RadiSys Corporation will not be liable for any damages including indirect or consequential, from use of the OS-9 operating system, Microware-provided software, or reliance on the accuracy of this documentation. The information contained herein is subject to change without notice.

## Reproduction notice

The software described in this document is intended to be used on a single computer system. RadiSys Corporation expressly prohibits any reproduction of the software on tape, disk, or any other medium except for backup purposes. Distribution of this software, in part or whole, to any other party or on any other system may constitute copyright infringements and misappropriation of trade secrets and confidential processes which are the property of RadiSys Corporation and/or other parties. Unauthorized distribution of software may cause damages far in excess of the value of the copies involved.

October 2001

# Table of Contents

**Chapter 1: Introduction** **5**

6 PersonalJava Solution for OS-9 Runtime Components
7 OS-9
8 Networking
8 SoftStax
8 LAN Communications Pak
8 Graphics
9 Multimedia Application User Interface (MAUI)
9 Window Manager
9 Application Framework
10 Java AWT
10 Java Virtual Machine (JVM)
11 Applications and Applets
11 Sample Applications
12 Additional Java Tools
12 Running Java On a Diskless System
13 Java Development Tools
14 Windows® JDK

**Chapter 2: Running PersonalJava Solution for OS-9** **15**

16 System Requirements
16 Host Hardware Requirements
16 Target Hardware Requirements
17 Installing PersonalJava Solution for OS-9
17 Installing the PersonalJava Solution for OS-9 files
17 Installing PersonalJava Solution onto the Host System
19 Installing Files onto the Target

19 Building the PersonalJava Demo Bootfile
21 Installing PersonalJava Solution on the Target Hardware
23 Running Java Applets
23 Run the loadjava script
25 Running an Applet
26 Considerations for Running Your Own PersonalJava Applications

**Appendix A: Java Load Script** **29**

30 Example Java Load Script
30 X86 loadjava Script

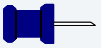
**Product Discrepancy Report** **33**

# Chapter 1: Introduction

This manual provides you with the information you need to get started with PersonalJava Solution for OS-9.

## For More Information

Refer to the current version of ***OS-9 Release Notes*** for possible last-minute updates to PersonalJava Solution for OS-9 or the x86 board.



## Note

Before proceeding, be certain you have installed either OS-9 for Embedded Systems or the OS-9 Board Level Solution (BLS) for your processor on your Windows-based host system. If you do not have either of these packages, contact your OS-9 supplier.

## For More Information

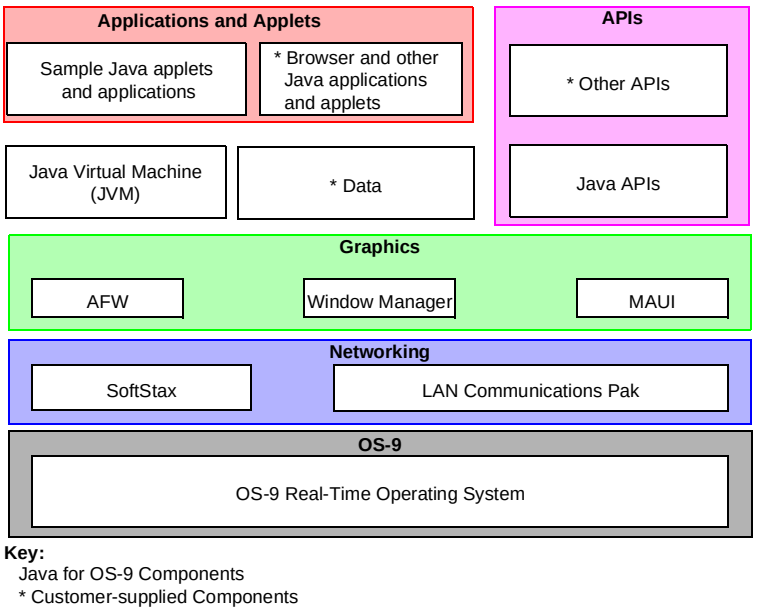
Refer to the CD-ROM insert for information about installing PersonalJava Solution for OS-9 on your Windows-based host platform.

# PersonalJava Solution for OS-9 Runtime Components

**PersonalJava Solution for OS-9** is a complete system software solution for developing Java-enabled devices. The PersonalJava Solution for OS-9 system consists of a scalable real-time operating system with specific software modules that help you create Java enabled devices without worrying about system software customization.

**Figure 1-1** shows the PersonalJava Solution for OS-9 architecture. Each software subsystem found in PersonalJava Solution for OS-9 is defined in the following sections.

**Figure 1-1 PersonalJava Solution for OS-9 Runtime Components**




**Key:**
Java for OS-9 Components
* Customer-supplied Components

### Note

Many of these components were installed with your **Enhanced OS-9 for X86** package. You must have installed either the **Embedded Systems** or **Board Level Solution** package prior to installing PersonalJava Solution for OS-9. Contact your OS-9 provider for a copy of one of these packages.

## OS-9

At the core of PersonalJava Solution for OS-9 is OS-9 and its support modules.

OS-9 is an architecturally advanced, high performance real-time operating system available for many microprocessor families. At its core is the OS-9 stand-alone microkernel.

Coupled with the power of the microkernel, the unique modular architecture of OS-9 enables dynamic loading of any OS-9 system or user application module while the system is up and running.

### For More Information

Refer to the OS-9 manual set for more information about the operating system.

## Networking

The ability to communicate with other computers or devices is essential for a Java-enabled device. PersonalJava Solution for OS-9 uses the standard SoftStax I/O implementation so a variety of transport layers can be used with Java.

### SoftStax

SoftStax provides a consistent application-level interface using a variety of networking protocols. The protocols necessary for using PersonalJava Solution for OS-9 are included in the LAN Communications Pak.

### LAN Communications Pak

The Microware LAN Communications Pak software consists of a TCP/IP protocol stack with UDP support, SLIP/CSLIP support, PPP support, and drivers for supported hardware.


**For More Information**

Refer to the SoftStax and LAN Communications Pak manual sets for more information about this implementation.

## Graphics

One of the strengths of Java as a programming language is its support for graphics. To handle graphics, PersonalJava Solution for OS-9 uses four components: MAUI, winmgr, the Application Framework (AFW), and the Java AWT.

## Multimedia Application User Interface (MAUI)

MAUI is a high-level library that manages the display of graphics, text, and user input.

### For More Information

Refer to the MAUI manual set for more information about the MAUI system.

## Window Manager

The PersonalJava Solution for OS-9 Window Manager (winmgr) is a MAUI application that manages windows. Three versions of the winmgr are available, each with different levels of functionality.

### For More Information

Refer to ***Using PersonalJava Solution for OS-9*** for more information about the Window Manager.

## Application Framework

The Application Framework (AFW) is a class library that contains the code necessary to display GUI components and handle events for an interactive application.

### Java AWT

PersonalJava Solution includes an Abstract Windowing Toolkit (AWT) package that allows Java applications to display GUI components, render images, draw graphics primitives, and respond to events. This package is standard across all PersonalJava implementations.

## Java Virtual Machine (JVM)

Consumer devices that interpret Java applications must contain the Java Virtual Machine (JVM). Java applications are comprised of Java classes consisting of byte codes.

Java byte codes are machine-independent and interpreted by the JVM. The purpose of the JVM is to interpret these Java byte codes and initiate appropriate actions on the host platform. In addition to executing byte codes in all classes within the system, the JVM also handles signals and Java exceptions, manages RAM, and manages threads.

# Applications and Applets

Along with the basic system components, Microware has included several sample applications and applets on the PersonalJava Solution for OS-9 CD.

## Sample Applications

Several sample applications have been included in this package. They are located in `MWOS\SRC\PJAVA\EXAMPLES`. Additional sample applets from Sun are located in `MWOS\DOS\jdk1.1.8\demo`.

# Additional Java Tools

## Running Java On a Diskless System

PersonalJava Solution for OS-9 includes a tool called the JavaCodeCompact (JCC) that enables sets of Java class files to be pre-loaded in RAM or placed in ROM. This is accomplished by pre-processing the class files into an assembly language file that is eventually turned into a module. The module can then be loaded at run-time at a pre-determined address or loaded into the ROM of the device. This process eliminates the need to have the class files themselves, often called `classes.zip`, resident on the device.


### For More Information

Refer to ***Using JavaCodeCompact for OS-9*** for instructions on using this tool in the OS-9 environment and refer to ***Using PersonalJava Solution for OS-9*** for information about creating Java applications for a diskless OS-9 target.

# Java Development Tools

Since Java is architecture neutral, users can develop their Java applications using any of the GUI-based Java development packages on the market. Some of these include Metrowerks CodeWarrior, Sunsoft's Java Workshop, and Symantec's Visual Cafe to name a few. As long as the output of the development environment is standard Java class files containing standard byte codes, the code is compatible with PersonalJava Solution for OS-9.

Standard Java class files contain a great deal of information about the source code from which they were compiled, including symbol names. With the appropriate tools, it is possible to de-compile Java code into an almost exact replica of the source code. Some of these tools address this problem by munging or obfuscating the object code so de-compilation is not as easy. Refer to the numerous Java-related web sites and UseNet news groups for information on these tools.

## For More Information

For more information about CodeWarrior, visit the Metrowerks website at `http://www.metrowerks.com/`.

For more information about Java Workshop, visit the Sun website at `http://www.sun.com/`.

For more information about Visual Cafe, visit the Symantec website at `http://www.Symantec.com/`.

# Windows® JDK

To make it easier for you to perform native method work, Microware has included the Windows Java Development Kit (JDK) v.1.1 in the package for the host system.

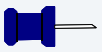
The `javah.exe` executable in this package has been modified to generate code that works with the Microware UltraC/C++ compiler.

The pre-loader classes are contained in the `jcc.zip` file. This file is on the Windows host machine in the `\MWOS\DOS\JDK1.1.8\lib` directory.

# Chapter 2: Running PersonalJava Solution for OS-9

This chapter explains how to install and run Microware's PersonalJava demo application and how to run your own Java applets and applications.

Microware's PersonalJava Solution can run in a disk based system or in a completely diskless environment. The examples used in this chapter assume you are installing PersonalJava Solution on a system that includes a standard IDE disk; however, the disk can be a PCMCIA ATA flash device, a SCSI disk, or a RAM disk loaded through an Ethernet connection using FTP.



**Note**

You must install the OS-9 OEM Package, or the OS-9 Board Level Solution before installing PersonalJava Solution for OS-9.

# System Requirements

The following represents the host and target requirements for running PersonalJava Solution for OS-9.

## Host Hardware Requirements

Your host PC should have the following hardware:

- Windows 95, 98, or NT
- 250-350 MB of free disk space
- 32MB of RAM
- an ethernet network card
- a terminal emulation program (such as Hyperterminal, which comes with Microsoft Windows 95/98 and NT)
- an available serial port
- a CD-ROM Drive

## Target Hardware Requirements

Your target X86/Pentium system requires the following hardware:

- a VGA or SVGA Supported Video Card (for use with MAUI)
- keyboard and mouse (for use with MAUI)
- PC Compatible Floppy and IDE Drives
- 16MB of RAM
- a supported ethernet network card
- an available serial port

# Installing PersonalJava Solution for OS-9

Installation of Java on the target is initiated on the PC and finished on the target. Therefore, the instructions are divided into the following headings:

- **Installing the PersonalJava Solution for OS-9 files**
- **Building the PersonalJava Demo Bootfile**
- **Installing PersonalJava Solution on the Target Hardware**
- **Running Java Applets**
- **Considerations for Running Your Own PersonalJava Applications**

## Installing the PersonalJava Solution for OS-9 files

PersonalJava Solution for OS-9 is first installed on the host system and then on the target X86/Pentium system.

### Installing PersonalJava Solution onto the Host System

Step 1. Insert the CD-ROM containing **PersonalJava Solution for OS-9 (X86)** into your CD-ROM drive.

Step 2. The installation menu should come up automatically. If the installation menu fails to appear, navigate to the Autorun directory on the CD-ROM and double click on Autrorun.exe.

Step 3. Select **PersonalJava Solution for OS-9 (X86)** from the setup menu.

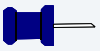
Step 4. Follow the directions in the installer windows. Enter the PersonalJava Solution for OS-9 password, when prompted. The password is located on the password card included with the PersonalJava Solution for OS-9 package.

Step 5. Enter the destination MWOS folder, when prompted. **Enhanced OS-9 for X86** must have been previously installed in this folder.

Step 6. Select the components to install, either PersonalJava Solution, PersonalJava Documentation or both.

Step 7. Select the program folder. By default, the package installs into **Enhanced OS-9 for X86**, some OS-9 packages created an **OS-9 for X86** program folder. Verify that you install Personal Java in the same folder as the previously installed OS-9 for X86 product.

Step 8. Click `Next` to complete the install.



**Note**

Be sure you are installing PersonalJava Solution for OS-9 into your MWOS directory tree. If you do not install PersonalJava Solution for OS-9 in your MWOS directory, PersonalJava Solution for OS-9 may not work correctly.

### Installing Files onto the Target

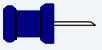
The files that go onto the target are found in the MWOS directory on the host machine. The path to the files is as follows: `MWOS\OS9000\80386\PORTS\<port name>\PJAVA.`

The `PJAVA` folder contains two relevant items: `pjava.mat` and `readme;txt`. `pjava.mat` is a Microware Archive Tool (MAT) archive of the files to go on the target and `readme.txt` explains how to install the MAT archive onto the your disk.

Refer to the following sections, **Building the PersonalJava Demo Bootfile** and **Installing PersonalJava Solution on the Target Hardware** for detailed installation instructions.

## Building the PersonalJava Demo Bootfile

This section discusses creating a disk-based boot for your X86 target hardware. It assumes that your target system is configured with a serial console, both floppy and IDE disk drives, VGA graphics and keyboard, a PS/2 style bus mouse, a supported network card and at least 16 MB of RAM. Refer to the **OS-9 for PCAT Board Guide** and the Wizard on-line help system for instructions on building a boot for other hardware options.



**Note**

You should create a new disk based boot, even if you already have a target system running OS-9 for X86.

**Note**

The JDK as shipped from Sun is targeted strictly at desktop environments. PersonalJava Solution for OS-9 may be used on either disk-based or diskless systems; the examples use a disk based system.

**Note**

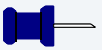
The OS-9 console must be moved to a serial port so that it does not conflict with the Java window manager running on the VGA/SVGA graphics hardware. This example assumes that the OS-9 console appears on COM1, and is connected to a host computer running a terminal emulation program such as Hyperterminal.

Follow the instructions in the **OS-9 for PCAT Board Guide** for detailed instructions on installing OS-9 on the target system and configuring a disk based boot image.

Once configured, your OS-9 target system should perform the following tasks:

- boot OS-9 from a hard disk
- bring the OS-9 system console up on a serial port, using an emulator running on the Windows host computer, such as hyperterminal
- have a working ethernet connection, tested by establishing a Telnet or FTP connection from the Windows host computer to the OS-9 target system
- have MAUI graphics configured on the target OS-9 system, tested by running the MAUI `fcopy` or `fdraw` demo programs on the target

# Installing PersonalJava Solution on the Target Hardware



## Note

The procedures in this stage assume the following things:

- your target system boots from the hard disk without error
- you have an OS-9 system prompt, using the terminal emulation program running on your Windows host computer
- you have an Ethernet connection between the host and target machines, you can telnet and FTP to the target, from the Windows host computer
- you have a supported graphics card and have installed the MAUI graphics system

Complete the following steps on the target computer.

Step 1. Use FTP to download the pjava tar file (`x86_pjava.tar`) from the `MWOS\OS9000\80386\PJAVA` directory on the host computer. Perform the following steps:

```
cd MWOS\OS9000\80386\PORTS\PCAT\PJAVA
ftp <target>
User: super
Password: user
ftp> bin
ftp> cd /h0
ftp> send pjava.mat
ftp> quit
```

Step 2. Untar the pjava image by typing the following commands on the OS-9 system console.

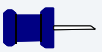
```
$ chd /h0/MWOS
$ load -d /h0/CMDS/mat
$ tmode nopause
$ mat -x -v ../pjava.mat
```

# Running Java Applets

This section describes what you need to do to prepare your target board to run the Sun Demo applets or your own applets. The Sun JDK v1.1 demo applets are contained in `MWOS/DOS/jdk1.1.7B/demo`.

## Run the loadjava script

The loadjava script sets up the OS-9 environment variables, loads the MAUI support modules into memory, initializes the modules, and runs the MAUI input process.



### Note

The loadjava script must be run every time the board is booted.

### For More Information

You can set up the loadjava script to run every time by defining it as a system startup script. See “Making a Startup File” in Chapter 9 of the ***Using OS-9*** manual.

Step 1. From the OS-9 console, change directories to the SYS directory by typing the following on the command line:

```
chd /h0/SYS
```

Step 2. Enter the following commands to run the loadjava script:

```
tmode nopause
profile loadjava
```

As the loadjava script executes, you should see a series of messages scroll up the screen.

Step 3. Type the following command to display the environment variables:

```
printenv
```

Step 4. Compare the listing on your screen with the following listing. Make sure that the listed environment variables are set correctly.

```
MWOS=/h0/MWOS
JAVA_HOME=/h0/MWOS/SRC/PJAVA
CLASSPATH=/h0/MWOS/SRC/PJAVA/LIB/classes.zip:.
PATH=/h0/MWOS/OS9000/80386/CMDS:$PATH
PORT=/term
HOME=/h0
USER=java_user
TZ=CST
```

Step 5. Make sure the `maui_inp` process is running by typing the following command:

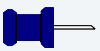
```
procs
```

You should see a listing of the processes that are currently running on your target computer. It should look similar to the following illustration.

```
Id PId Grp.Usr  Prior  MemSiz Sig S    CPU Time   Age Module & I/O
  2   0   0.0     128   52.00k   0 w        0.28   ??? mshell <>>>term
  3   2   0.0     128   68.00k   0 s        0.05   ??? telnetd <>>>nil
  4   0   0.0     128   20.00k   0 e        0.00   ??? spf_rx
  5   2   0.0     128   56.00k   0 s        0.07   ??? ftpd <>>>nil
  6   2   0.0     256   24.00k   0 s        0.01  0:00 maui_inp <>>>term
  7   2   0.0     128   48.00k   0 *        0.03  0:00 procs <>>>term
```

Step 6. Check that `maui_inp` is listed under the module heading.

You are now ready to run your applet.

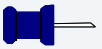


**Note**

If your applet requires resources that are not present on your target hardware (sound for example), then it may not work correctly.

## Running an Applet

The best way to test the PJava installation on the OS-9 target is to download and run an applet. Several example applets were installed in the `\MWOS\DOS\jdk1.1.7B\demo` directory on the Windows host. This example downloads and runs the Jumping Box applet.



### Note

The PersonalJava Solution Window Manager must be started before running this example applet. At the OS-9 console, type: `winmgr ^250 <>>>/nil&`

Step 1. Change to the JumpingBox demo directory on the Windows computer.

```
cd \mwos\dos\jdk1.1.7B\demo\Fractal
```

Step 2. Use FTP to download the Jumping box example files.

```
ftp <target>
User: super
Password: user
ftp> bin
ftp> cd /h0
ftp> send example1.html
ftp> prompt
ftp> mput *.class
ftp> quit
```

Step 3. Run the applet from the OS-9 console by typing the following:

```
chd /h0
appletviewer example1.html
or
pjava sun.applet.AppletViewer example1.html
```

# Considerations for Running Your Own PersonalJava Applications

The loadjava script took care of a number of details that you should be aware of when running your own applications. The following section is a complete list of details that need to be addressed.

## For More Information

See **Appendix A: Java Load Script** for an example loadjava script.

If your ultimate target is a diskless system, then the steps taken in loadjava have to be accomplished by setting the environment variables in the init module and including the loaded modules in the bootfile/ROM image.

The environment variables need to be set correctly for the target system. Omit environment variables that are not applicable (e.g. for a diskless environment, variables set to disk paths need not be set). These include the following items:

- MWOS–location of your MWOS directory
- JAVA_HOME–location of your Java properties files
- CLASSPATH–list of directories and zip files to search for class files
- LD_LIBRARY_PATH–list of directories to search for native method libraries
- PATH–list of directories to search for executable files
- PORT–device used to communicate with the user
- USER–name used to refer to the user
- HOME–home directory for the user
- TZ–time zone setting for the system

The modules (executable code and configuration data) for PersonalJava Solution need to be in memory or at their appropriate location on the disk if applicable. These include the following modules:

`winmgr`–PersonalJava Window Manager (alternatively, `winmgrs` or `winmgrg` could be used)

`winmgr.dat`–Window Manager settings

`stock_8.res`–8-bit image resources for the Window Manager and application framework

`stock_9.res`–16-bit image resources for the Window Manager and application framework

`pjava`–PersonalJava Solution

`libmawt.so`–AWT shared library module

`libmawt_0.dat`–pre-computed color palette

font modules–various modules related to font rendering

MAUI modules–various modules related to graphics support

`*.properties`–properties files needed by PersonalJava Solution (e.g. from a modman archive)

Before running a graphical PersonalJava application, the Window Manager must be started. Before running the window manager, the MAUI input process must be started. The following command lines show an example startup sequence:

- `maui_inp ^255 <>>>/nil &`
- `winmgr ^200 <>>>/nil &`
- `pjava <application class>`

Examine the system running the demos and the loadjava script for more information on configuring a system to run your own PersonalJava applications.

# Appendix A: Java Load Script

This appendix lists the Java Load Script needed to start Java on your target platform.

# Example Java Load Script

This load script was used to set up the OS-9 init module to run Java. Use this script as a basis for your own scripts when configuring your system to run Java applications.

## X86 loadjava Script

The following example LoadJava script is used on a PC-AT target.

```
-tnp
* * * * * * * * * * * * * * * * * * * * *
* Load script for PersonalJava v3.0.1
*


*
* Set environment variables
*
setenv MWOS /dd/MWOS
setenv JAVA_HOME /dd/MWOS/SRC/PJAVA
setenv CLASSPATH
/dd/MWOS/SRC/PJAVA/LIB/classes.zip:.
setenv PATH /dd/MWOS/OS9000/80386/CMDS:$PATH
setenv LD_LIBRARY_PATH
/dd/MWOS/OS9000/80386/LIB/SHARED
setenv PORT /term
setenv HOME /dd
setenv USER java_user
setenv TZ CST

*
* Load the Java Window Manager application and a file
containing
* the resource module ‘Stock.res’
*
```

```
*  winmgrs- Simplest/smallest Window Manager - No
window frames
*  winmgr- Default Window Manager
*  winmgrg- Debug version - “Send Shutdown Message” &
“Dump Window Tree” fuctionality
*
*  Stock_8.res - 8-bit bitmap and cursor support
(default)
*  Stock_9.res - 16-bit bitmap and cursor support
(PCAT only)
*
let winmgr = “winmgr”;* winmgr, winmgrg, or winmgrs
load -d /dd/MWOS/OS9000/80386/CMDS/%winmgr
*
load -ld
/dd/MWOS/OS9000/80386/PORTS/PCAT/CMDS/stock_8.res
*load -ld
/dd/MWOS/OS9000/80386/PORTS/PCAT/CMDS/stock_9.res


*
* Load the Window Manager Settings
load -ld /dd/MWOS/OS9000/80386/CMDS/winmgr.dat

*
* Load the pre-generated libmawt Color Cube module
load -ld
/dd/MWOS/OS9000/80386/PORTS/PCAT/LIB/SHARED/libmawt_
0.dat

*
* Load fonts for “font.properties”
*
load -d /dd/MWOS/OS9000/80386/ASSETS/FONTS/AGFA/MT/*
load -d
/dd/MWOS/OS9000/80386/ASSETS/FONTS/AGFA/TT/utt.ss

*
```

```
* Initialize the keyboard and mouse devices
*
iniz m0 k0

*
* The two lines below can be removed if non-graphical
Personal Java applications
* are to be used.

*
* Launch the MAUI input process
*
maui_inp ^256 <>>>/nil &

*
* Start window manager loaded above
*
%winmgr ^250 <>>>/nil &

-nt
```

# Product Discrepancy Report

To: Microware Customer Support

FAX: 515-224-1352

From:_____

Company:_____

Phone:_____

Fax:_____Email:_____

Product Name:

Description of Problem:

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

Host Platform_____

Target Platform_____

RadiSys.
MICROWARE SOFTWARE