1 Overview

The purpose of this project was to create a lightweight system monitoring utility specifically for Windows XP.

1.1 Key Features

- Small, Easy to Use Simple.
- Provide Access to as much information as possible.
- Easily expandable, to allow for the addition of more information.
- Themeable, to provide a customizeable look and feel.

2 Technology Considerations

In making a decision as to what language, and utilities to use initial consideration was given solely to ease of development. Secondary consideration was given to the learning aspect of this project and the ability for it to provide an opportunity to explore different areas and methods for coding this project. With these considerations in mind it was decided that C# would be the language of choice for this project so that we could make full and easy use of the .NET framework. These tools have in this particular case made coding and design of this project simple and straightforward in many ways.

2.1 C# and .NET Overview

The first information nescessary when coding a C# windows project is a basic layout of the way .NET handles windows. Under C#.Net a window is a form, which can contain any number of controls. These controls are things such as a menu, or dialog box etc. Then the only other oddities that need to be introduced are properties. Basically what a property is is a replacement for the common get and set functions that classes tend to have whenever they want to protect access to a certain variable, they have the added benefit though of being used automagically whenever you write to or read from the property.

2.2 Technology Problems

Unfortunately it doesn't quite fit the small requirement memory wise, the smallest C# window that I have made has taken 10M and this project requires approx 17M to run.

3 Design and Project Architecture

The project is a fairly simple Form which contains any number of monitors, which are simply a panel. These monitors are then associated with a performance counter which will pull the information from the datasource and then update the monitor with the new data. To accomplish the update task each of the performance counters has a thread assigned to it that will pull the data at a regular interval. To allow for different monitor display types each monitor inherits from a base class which simply contains information on how to add, remove, and setup all of the monitors.